System for performing intelligent analysis and segmentation of a computer database

Patent Number: EP0840240

Publication date: 1998-05-06

Inventor(s): COPAS KEVIN WAYNE (US); LINDSAY MARSHALL PAUL (US); LUDWIG JEFFREY PAUL (US); SCHUBERT RICHARD NEAL (US); WIKLE GLENN KEITH (US); TOHERI SOHEILA (US); ANAND TEJWANSH SINGH (US); COULTER SCOTT DALE (US); KNUTSON JAMES FOSTER (US); LETTINGTON DREW THOMAS (US)

Applicant(s): NCR INT INC (US)

Requested Patent: JP10207751

Application Number: EP19970308374 19971021

Priority Number(s): US19960742006 19961031; US19960742007 19961031

IPC Classification: G06F17/30

EC Classification: G06F17/30B

Equivalents:

Cited Documents:

Abstract

A system for performing intelligent analysis, segmentation and partition of a database based upon attributes associated with the data in the database are provided. A report may be generated which allows a user to make decisions, without requiring the user to understand or interpret data itself. A database computer includes a database containing the data. The data includes a collection of information about an enterprise of the user. A server computer is coupled to the database computer and executes a database management program. A client computer is coupled to the server and executes an application program. The application program allows a user to define predetermined data types, to define relationships between the data types, to define parameters for the report, to define a method of analysis for the report, and to create the report. The report summarizes the data in terms of the data types, the data relationships and the method of analysis.

A1

Citation 1

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－207751

(43)公開日　平成10年(1998)８月７日

(51)Int.Cl.[6]	識別記号	ＦＩ		
Ｇ０６Ｆ　12/00	５１２	Ｇ０６Ｆ	12/00	５１２
17/30			15/40	３１０Ｃ
			15/401	３２０Ｚ
			15/403	３６０Ｚ

審査請求　未請求　請求項の数３　書面　外国語出願　（全288頁）

(21)出願番号　特願平9－334746

(22)出願日　平成９年(1997)10月29日

(31)優先権主張番号　０８／７４２，００６
(32)優先日　1996年10月31日
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０８／７４２，００７
(32)優先日　1996年10月31日
(33)優先権主張国　米国（ＵＳ）

(71)出願人　592089054
エヌシーアール　インターナショナル　インコーポレイテッド
ＮＣＲ　Ｉｎｔｅｒｎａｔｉｏｎａｌ，Ｉｎｃ．
アメリカ合衆国　45479　オハイオ、デイトン　サウス　パターソン　ブールバード　1700

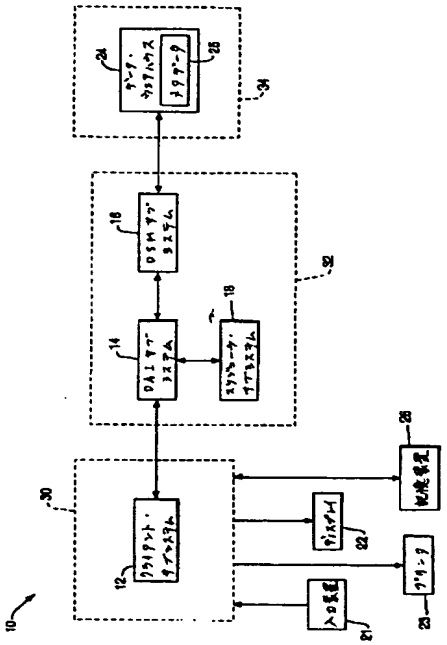
(72)発明者　テジュワンシュ　シン　アナンド
アメリカ合衆国　ジョージア州　30075　ロスウェルシャドウブルク　ドライブ　145

(74)代理人　弁理士　西山　善章

最終頁に続く

(54)【発明の名称】　データ管理システム

(57)【要約】
【課題】　グラフィック・データを、サーバ・コンピュータからクライアント・コンピュータに送信し、ユーザに表示するハイパーテキスト・データ処理システムに関する。
【解決手段】　ハイパーテキスト・マークアップ言語（ＨＴＭＬ）への拡張を使用して、図形的データがサーバ・コンピュータバからクライアント・コンピュータへ送られるハイパーテキスト・データ処理システム。クライアント・コンピュータは、図形的データの文法関係を解析し、表示対象のグラフを表す対象物を作成する。上記対象物は、グラフを表示するグラフ・サーバへ送られる。



【特許請求の範囲】
【請求項1】　データベース管理システムであって、
（1）関連付けられた属性を有するデータ・エンティティを含んでいるデータベースと、
（2）前記データベースに結合されているサーバと、
（3）前記サーバに結合されているクライアント・コンピュータとを備え、
前記データ・エンティティが第1階層の中で並べられていて、前記第1階層の各レベルが少なくとも1つの属性に関連付けられていて、
前記サーバ・コンピュータが、
（a）選択された属性を制限するための属性制限値を受信するための受信手段と、
（b）前記データベースをセグメント化するためのセグメント化手段とを含み、その中で第2階層が生成され、前記第2階層が前記属性制限値と関連付けられていて、
前記クライアント・コンピュータが、（a）前記属性制限値を前記クライアント・コンピュータのユーザから入力するための手段と、（b）前記属性制限値を前記受信手段に対して送信するための手段とを含むことを特徴とするデータベース管理システム。
【請求項2】　データベース管理システムであって、
（1）1つの組織を示しているデータを含んでいるデータベースと、
（2）前記データベースに対して結合されているサーバ・コンピュータと、
（3）前記サーバに対して結合されているクライアント・コンピュータとを含み、
前記サーバ・コンピュータが、
（a）定義されたトリガ・イベントの発生に応答して、あるいは前記データベースの中の前記データを分析するための要求に応答して、前記データベースに対して問い合わせるための問合わせ手段と、
（b）前記問合わせ手段に対して応答するレポートを作成するためのレポート作成手段とを含み、
前記クライアント・コンピュータが、（a）前記レポート作成手段によって作成される前記レポートを受信するための手段と、（b）前記クライアント・コンピュータのユーザに対して前記レポートを表示するための手段とを含んでいることを特徴とするデータベース管理システム。
【請求項3】　データベース管理システムであって、
（1）1つの組織を示すデータ・エンティティを含んでいるデータベースと、
（2）前記データベースに対して結合されているサーバ・コンピュータと、
（3）前記サーバに対して結合されているクライアント・コンピュータとを含み、
前記データ・エンティティが第1階層の中に並べられていて、前記第1階層の各レベルが少なくとも1つの属性に関連付けられており、
前記サーバ・コンピュータが、
（a）定義されたトリガ・イベントの発生に応答して、あるいは前記データベースの中の前記データを分析するための要求に応答して、前記データベースに対して問い合わせるための問合わせ手段と、
（b）前記問合わせ手段に対する応答のレポートを作成するためのレポート作成手段と、
（c）選択された属性を制限するための属性制限値を受信するための受信手段と、
（d）前記属性制限値に関連付けられている第2階層を生成するために前記データベースをセグメント化するためのセグメント化手段とを含み、
前記クライアント・コンピュータが、（a）前記属性制限値を前記クライアント・コンピュータのユーザから入力するための手段と、（b）前記属性制限値を前記受信手段に対して送信するための手段と、（c）前記レポート作成手段によって作成された前記レポートを受信するための手段と、（d）前記レポートを前記クライアント・コンピュータのユーザに対して表示するための手段とを含むことを特徴とするデータベース管理システム。
【発明の詳細な説明】
【0001】
【発明の属する技術分野】本発明はエキスパート・システムおよびレポート・システムに関し、特にコンピュータのデータベースを分析してセグメント化するため、およびレポートを作成するためのデータベース管理システムに関する。
【0002】
【従来の技術、及び、発明が解決しようとする課題】多くのアプリケーションにおいて、大量のトランザクション・レベルのデータが後で分析するために格納されている（データ・ウェアハウス）。データ・ウェアハウスを使うためには、そのデータが検索され、編成されてから、理解できる形式で示されなければならない。データ・ウェアハウスからデータを検索し、分析し、そして提示するために発見ツールが使われる。これらのツールとしては、非常に複雑なモデリング・ツールから、ユーザからＳＱＬデータベース・プログラミング言語の複雑性をマスクする以外には何もしないように設計されている比較的単純なエンド・ユーザ問合わせツールまでの範囲のものがあり得る。傾向または関係を求めてデータをサーチする自動化されたツールも発見ツールと考えられる。
【0003】市場には各種のツール・ベンダがあり、それらのベンダの製品は知識発見プロセス全体のうちの一部に対するソリューションを提供する。したがって、それらのデータを効果的に利用するためには、ユーザ・コミュニティは複数の分離されたツールを使うことを余儀なくされている。さらに、これらのツールはそのツール

の中に実装されているデータおよびデータベースのフォーマットまたは各種の分析方法についての深い知識を有する専門家のユーザを対象として作られている。また、既存の製品はユーザに明示的にそして繰返し的に知識を表現させない。最後に、既存のツールの出力はユーザが分析して解釈しなければならない複数のテーブルから構成されている。

【０００４】データ・ウェアハウス、そして一般にデータベースは、データの検索を効率化するために複雑な構造を有しており、それらをエンド・ユーザが使うのは簡単ではない。ユーザはデータベースの言葉ではなく、ユーザの言葉でのレポートを望んでいる。市場にあるいくつかのツールによって、ユーザは新しい用語を定義してそれらの用語をデータベースにマップすることができるが、新しい用語の関連付けられた集合の管理はサポートされない。すなわち、既存の用語に対する新しい用語の関係は自動的にはユーザに対しては検出されない。これらの問題点の他に、レポートの内容についてユーザが別の、同様なレポートを要求することになるのが普通である。関連付けられたレポートの集合の記憶および再使用（新しいデータの集合についてそのレポートを再生成すること）も望まれる。新しいデータについての関連付けられたレポートの生成およびそのレポートの再生成は市場では十分には利用できない。

【０００５】ユーザがデータベースの底流にあるデータ構造についての、あるいはデータベースのプログラミング言語についての知識を必要とせずに、１つのツールでデータを検索して分析することができ、ユーザが新しい用語を定義して用語間の関係を検出して管理することができ、ユーザが簡単に関連のレポートを生成することができ、そして新しいデータについての関連付けられたレポートの集合を再生成できるような、コンピュータ・データベースからのレポートを生成するためのシステムを提供することが１つの望ましい目的である。本発明のもう１つの目的はユーザがデータベースの中のデータに関連付けられている属性に基づいてデータベースをセグメント化して区画化することができるシステムを提供することである。

【０００６】

【課題を解決するための手段】本発明の第１の態様によれば、次のものを特徴とするデータベース管理システムが提供される。

（１）関連付けられた属性を有するデータ・エンティティを含んでいるデータベースと、

（２）前記データベースに結合されているサーバと、

（３）前記サーバに結合されているクライアント・コンピュータとを備え、前記データ・エンティティは第１階層の中で並べられていて、前記第１階層の各レベルが少なくとも１つの属性に関連付けられていて、前記サーバ・コンピュータが、

（ａ）選択された属性を制限するための属性制限値を受信するための受信手段と、

（ｂ）前記データベースをセグメント化するためのセグメント化手段とを含み、その中で第２階層が生成され、前記第２階層は前記属性制限値と関連付けられていて、前記クライアント・コンピュータが、（ａ）前記属性制限値を前記クライアント・コンピュータのユーザから入力するための手段と、（ｂ）前記属性制限値を前記受信手段に対して送信するための手段とを含んでいる。

【０００７】本発明の第２の態様によれば、次のことを特徴とするデータベース管理システムが提供される。

（１）１つの組織を示しているデータを含んでいるデータベースと、

（２）前記データベースに対して結合されているサーバ・コンピュータと、

（３）前記サーバに対して結合されているクライアント・コンピュータとを備え、前記サーバ・コンピュータが、

（ａ）定義されたトリガ・イベントの発生に応答して、あるいは前記データベースの中の前記データを分析するための要求に応答して、前記データベースに対して問い合わせるための問合わせ手段と、

（ｂ）前記問合わせ手段に対して応答するレポートを作成するためのレポート作成手段とを含み、前記クライアント・コンピュータが、（ａ）前記レポート作成手段によって作成される前記レポートを受信するための手段と、（ｂ）前記クライアント・コンピュータのユーザに対して前記レポートを表示するための手段とを含んでいる。

【０００８】本発明のさらにもう１つの態様によれば、次のものを含むデータベース管理システムが提供される。

（１）１つの組織を示すデータ・エンティティを含んでいるデータベースと、

（２）前記データベースに対して結合されているサーバ・コンピュータと、

（３）前記サーバに対して結合されているクライアント・コンピュータとを備え、前記データ・エンティティが第１階層の中に並べられていて、前記第１階層の各レベルが少なくとも１つの属性に関連付けられており、前記サーバ・コンピュータが、

（ａ）定義されたトリガ・イベントの発生に応答して、あるいは前記データベースの中の前記データを分析するための要求に応答して、前記データベースに対して問い合わせるための問合わせ手段と、

（ｂ）前記問合わせ手段に対する応答のレポートを作成するためのレポート作成手段と、

（ｃ）選択された属性を制限するための属性制限値を受信するための受信手段と、

（ｄ）前記属性制限値に関連付けられている第２階層を

生成するために前記データベースをセグメント化するためのセグメント化手段とを含み、前記クライアント・コンピュータが、（ａ）前記属性制限値を前記クライアント・コンピュータのユーザから入力するための手段と、（ｂ）前記属性制限値を前記受信手段に対して送信するための手段と、（ｃ）前記レポート作成手段によって作成された前記レポートを受信するための手段と、（ｄ）前記レポートを前記クライアント・コンピュータのユーザに対して表示するための手段とを含んでいる。

【０００９】

【発明の実施の形態】ここで図１を参照すると、システム１０が４つの主要サブシステム、すなわち、クライアント・サブシステム１２、データ抽象化インテリジェンス（ＤＡＩ）サブシステム１４、データおよびスキーマ操作（ＤＳＭ）サブシステム１６、およびスケジューラ・サブシステム１８を含んでいる。システム１０の説明に関連して、次の定義が用意されている。「アラート条件（Ａｌｅｒｔ　Ｃｏｎｄｉｔｉｏｎ）」はその条件が満足された時にアラート・メッセージを返すユーザ定義の１つの条件または一組の条件である。例えば、アラート条件はブランドＡのシャツの在庫量が、ある週において２００ユニット以下に落ちた時、システム１０はアラート・メッセージ、ＩｎｆｏＦｒａｍｅ™を作成するか、あるいは別の分析を行う。

【００１０】「アラート・メッセージ（Ａｌｅｒｔ　Ｍｅｓｓａｇｅ）」はアラート条件が満足されたことをユーザに通知するメッセージである。アラート・メッセージから、ユーザは作成されるべき対応している情報レポート（ＩｎｆｏＦｒａｍｅ™）を選択することができる。アラート・メッセージの一例は「警告：ブランドＡのシャツの在庫が２００以下になっています（Ａｌｅｒｔ：ｔｈｅ　ｉｎｖｅｎｔｏｒｙ　ｏｆ　ｂｒａｎｄ　Ａ　ｓｈｉｒｔｓ　ｉｓ　ｂｅｌｏｗ　２００）」である。「アラート情報レポート（Ａｌｅｒｔ　ＩｎｆｏＦｒａｍｅ™）」はアラート・メッセージを詳細に記述している一種のステータス・レポートである。Ａｌｅｒｔ　ＩｎｆｏＦｒａｍｅ™は何が起こったか、いつ起こったか、そしてそれが何故起こったかの理由であると考えられることを記述している。「アナリスト（Ａｎａｌｙｓｔ）」はアラートをトリガしなければならないデータの中の１つのイベントを規定する。あるいは、１つのＩｎｆｏＦｒａｍｅ™の中でレポートされるべき分析および測定項目およびセグメントのタイプを規定し、そしてオプションとしてこのＩｎｆｏＦｒａｍｅ™が作成されるスケジュール、あるいはＩｎｆｏＦｒａｍｅ™をトリガしなければならないデータの中のそのイベントのタイプを規定する。

【００１１】「属性（ａｔｔｒｉｂｕｔｅ）」はウェアハウスの中で表される１つのエンティティの特性または特徴である。例えば、色（Ｃｏｌｏｒ）、製造者（Ｍａｎｕｆａｃｔｕｒｅｒ）、またはサイズ（Ｓｉｚｅ）は衣類（Ｃｌｏｔｈｉｎｇ）の製品カテゴリーのすべての属性である。「属性の制限（ａｔｔｒｉｂｕｔｅ　ｒｅｓｔｒｉｃｔｉｏｎ）」は属性が持ち得る値を制限する表現である。例えば、「価格が＄１０．００より安い」の中での「＄１０．００より安い」は「価格」属性についての１つの制限である。もう１つの例は「女性用の衣類または男性用の衣類」は「部門」属性における制限である。「青（Ｂｌｕｅ）」や「男性用の衣類」などの単独の属性値も１つの属性制限である。

【００１２】データ・ウェアハウスの中の特定のエンティティ（製品など）は一組の「属性」および「値」として表される。例えば、製品「デザイナーＸ男性用シャツ、サイズ４２、色は青」は「製品：部門：男性用衣類；製造者：デザイナーＸ；サイズ：４２；色：青」として表される。これらの値は各属性に対する特定のドメインのメンバである（下記参照）。

【００１３】「インジケータ（Ｉｎｄｉｃａｔｏｒｓ）」は通常は数値に関連付けられる概念についての分類（例えば、売上げ数量、在庫、価格）である。インジケータはそれぞれの計算に関係する方法および式（例えば、総売上高）およびインジケータ間の因果関係（例えば、価格が増加した場合、売上げ数量が減少する筈である）を有する。１つのインジケータの中でセグメントを定義することができ、それは対象とする特定のグループのインジケータ（例えば、年配の顧客、Ａ社の製品）を記述する。

【００１４】「変動分析レポート（Ｃｈａｎｇｅ　Ａｎａｌｙｓｉｓ　Ｒｅｐｏｒｔ）」は２つの期間にわたってインジケータを説明している複合ドキュメントである。システム１０の内部では、２つの期間を指定し、その期間に対する選定されたインジケータの差を見ることができる（例えば、「今年の繊維製品の売上げは去年の売上げに比べてどうか？」）。変動分析レポートは日、週、月、四半期、年、または他の定義された期間に対する結果をレポートすることができる。

【００１５】「比較分析（Ｃｏｍｐａｒｉｓｏｎ　Ａｎａｌｙｓｉｓ）ＩｎｆｏＦｒａｍｅ™」は、同じ期間にわたっての２つのインジケータの値をユーザが比較すること、あるいは同じ期間にわたっての２つの兄弟セグメントについての同じインジケータの値をユーザが比較するのを支援する一種のステータス・レポートである。「複合インジケータ（Ｃｏｍｐｏｕｎｄ　Ｉｎｄｉｃａｔｏｒ）」はプリミティブのインジケータを算術および集合演算で組み合わせることによって作られるユーザ定義のインジケータである。

【００１６】「データ・ウェアハウス（Ｄａｔａ　Ｗａｒｅｈｏｕｓｅ）」は１つまたはそれ以上のデータベースの中に収容されるデータの非常に大きな集合である。これらのデータベースは普通はデータベースのサーバ・

コンピュータ上にあり、1つのロケーションまたは地理的に分散された場所のいずれかに存在する可能性がある。「ディメンション（Ｄｉｍｅｎｓｉｏｎ）」はエンティティの高レベルのカテゴリーである。例えば、「小売り」のドメインにおいて、ディメンションは「製品」、「マーケット」および「時間」（時間は任意のドメインに対して適用可能な汎用のディメンションである）などがある。ディメンションはそのエンティティを記述するために使うことができる一組の属性に関連付けられている。例えば、「ブランド」、「製造者」および「サイズ」は1つの製品のディメンションを記述する。

【００１７】すべての属性は値の暗黙の、あるいは明示的な「ドメイン」を持っている。例えば、「部門」属性に対する値のドメインはその企業に対する正式な部門の符号化であり、「サイズ」属性に対する値のドメインは規定された単位でそのサイズを表している非ゼロの数値である。

【００１８】「ドリル・ダウン・ヒューリスティック（Ｄｒｉｌｌ　Ｄｏｗｎ　Ｈｅｕｒｉｓｔｉｃ）」はユーザに対してレポートされなければならないセグメントのフリー属性セグメントの測定項目値間の何らかの関係を規定する。「エンド・ユーザ（Ｅｎｄ　Ｕｓｅｒｓ）」はシステム１０が特にそのユーザを対象として設計されているユーザである。エンド・ユーザはユーザの操作についての知識を普通に有し、そしてこの例の場合はＭｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ™（Ｗｉｎｄｏｗｓ　３．１™、Ｗｉｎｄｏｗｓ　ＮＴ™およびＷｉｎｄｏｗｓ　９５™など）を使っている。エンド・ユーザは普通は自分がアクセスしたいデータベースの詳しいデータ構造またはＳＱＬのコード作成についての専門性を持たない。

【００１９】「Ｅｎｔｅｒｐｒｉｓｅ　Ｉｎｆｏｒｍａｔｉｏｎ　Ｆａｃｔｏｒｙ™（企業情報ファクトリ）」（ＥＩＦ）は代表的なユーザがデータ・ウェアハウスのそれぞれのデータにアクセスすることができる商用のソフトウェア・パッケージである。データ・ウェアハウスは本質的に受動的な環境であり、普通はデータにアクセスするためのＳＱＬコードの使用およびデータベースの構造についての知識を必要とする。ＥＩＦはユーザがＳＱＬまたはデータベースの知識なしに、それぞれのデータベースからデータを取り出すことができるようにするツールを提供するための基礎を提供しているという点でデータ・ウェアハウスとは異なっている。

【００２０】「例外アナリスト（Ｅｘｃｃｐｔｉｏｎ　Ａｎａｌｙｓｔ）」はトリガ条件をテストするために定期的に実行されるアナリストであり、そしてそのトリガ条件が発生した時にアラートまたはレポートを返す。属性のドメインが有限の集合（上記の「部門」のような）であった場合、それは有限ドメインと呼ばれる。それに対するものは価格（Ｐｒｉｃｅ）などの無限または連続のドメインである。

【００２１】「自由属性（Ｆｒｅｅ　Ａｔｔｒｉｂｕｔｅ）」は1つのセグメントの属性であって、そのセグメントを定義するように制限されていない属性である。色（Ｃｏｌｏｒ）、コスト（ｃｏｓｔ）、および重量（Ｗｅｉｇｈｔ）はすべてセグメント「高価なシャツ（Ｅｘｐｅｎｓｉｖｅ　Ｓｈｉｒｔｓ）」の自由属性となり得る。「ヒューリスティック・ルール（Ｈｅｕｒｉｓｔｉｃ　Ｒｕｌｅ）」はアナリストによって見つけられたいくつかのデータの条件、そのセグメントの測定項目値間のいくつかの関係を規定し、それは完成されたＩｎｆｏＦｒａｍｅ™の中でユーザに対してレポートされ得る必要がある。

【００２２】「ハイパーテキスト・マークアップ言語（ＨｙｐｅｒＴｅｘｔ　ＭａｒｋｕｐＬａｎｇｕａｇｅ）」（ＨＴＭＬ）はテキスト・ドキュメントの中にハイパーリンクおよびグラフィッタ（絵、グラフ、表）を含めることができるソフトウェア・ドキュメントのための、最近使われるようになってきた標準フォーマットである。ハイパーリンクはそのドキュメントの中の「ホット」領域（普通はその回りのテキストとは異なる色のテキスト）であり、その上でクリックされると、元のＨＴＭＬドキュメントに関連付けられているか、あるいはリンクされている別のドキュメントを表示する。

【００２３】「ＩｎｆｏＦｒａｍｅ™定義（Ｄｅｆｉｎｉｔｉｏｎｓ）は」は特定の概念、インジケータ、時間間隔、分析タイプ、およびセグメントを含めるためにカスタマイズされているシステム・テンプレートである。ＩｎｆｏＦｒａｍｅ定義は1つの「ＩｎｆｏＦｒａｍｅ™」を作るためにただちに「実行する」ことができ、後で実行されるために保存されるか、あるいは後で実行されるために保存されてスケジュールされるか、あるいは別のアナリストによってトリガされる。

【００２４】「知識作業者（Ｋｎｏｗｌｅｄｇｅ　Ｗｏｒｋｅｒ）」はデータ・ウェアハウスの構造を知っていて、分析のバックグラウンドを有する、ＳＱＬについてよく知っている人である。さらに、知識作業者は統計的およびデータ分析のパッケージおよびデータ・モデリング・ツールを使うことができる。「Ｍａｎａｇａｍｅｎｔ　Ｄｉｓｃｏｖｅｒｙ　Ｔｏｏｌ™（管理発見ツール）（ＭＤＴ）」は本発明の総合的なシステムを指す。

【００２５】「Ｍｅｔａｄａｔａ™（メタデータ）」はエンド・ユーザの特定の操作に関する情報の集まりであり、コア、パブリックまたはプライベートの3種類のうちの1つであり得る。インストレーションの後、この情報はそのエンド・ユーザのデータベースの中に格納され、そしてそのエンド・ユーザの特定のニーズに合わせてレポートを作成するために使われる。Ｍｅｔａｄａｔａ™は概念、インジケータ、セグメント、属性、属性の値および測定項目間の関係を含むが、それらに限定さ

れるわけではない。

【0026】MetadataTMのコア・セットは専門のサービス員およびMDTアドミニストレータによってインストレーション時にセットアップされるのが普通である。コアのMetadataTMはディメンション、属性、基本測定項目、セグメントおよび年の定義から構成される。パブリックのMetadataTMはMDTアドミニストレータおよび知識作業者のユーザ・タイプによってのみ変更可能であり、インストレーション後に定義されて修正される。パブリックのMetadataTMはパブリックの複合測定項目、パブリックの測定項目間の関係およびパブリックのセグメントを含む。

【0027】プライベートのMetadataTMは各ユーザに所属しており、エンド・ユーザ（エグゼクティブ・ユーザ）のユーザ・タイプによってのみ変更可能である。プライベートのMetadataTMはプライベートの複合測定項目、プライベートの測定項目間の関係およびプライベートのセグメントを含む。属性のドメインが有限である場合、「自然パーティション（NaturalPartition）」は各セグメントがそのドメインの各要素に対応するようなパーティションである。例えば、部門の属性の自然パーティションはセグメント「男性用の衣類」、「婦人の衣類」などの集合である。

【0028】「オブジェクト・リンキングおよびエンベッディング（Object Linking and Embedding）（OLE）」はコンピュータのドキュメントの中のオブジェクト（例えば、グラフ、表）が、その上でクリックされた時にそのオブジェクト（グラフ、表、ドキュメント）を生成したソフトウェア・アプリケーションを立ち上げることができるコンピュータ・フォーマットである。与えられたパーティションに対するユーザ定義のセグメントがそのドメインをカバーしない場合、「その他（Other）」のセグメントがそのパーティションの残りの部分を代表することになる。

【0029】「パーティション（Partition）」は単独の属性の制限によるディメンションの暗黙の、あるいは明示的な分割である。例えば、価格の属性、およびその「$10．00より安い」という制限を取り上げると、これは2つの集合またはセグメント、すなわち、価格が10ドルより安い製品、および価格が10ドルより高いか、あるいはそれに等しい製品に、製品を区画化することを定義する。パーティションの集合またはセグメントは元の集合をカバーしなければならならず、そしてオーバラップしてはならないことに留意されたい。すなわち、「価格＜$10．00」および「価格＜$15．00」はパーティションを定義しない。

【0030】「プリミティブ・インジケータ（Primitive Indicators）」はデータ・ウェアハウスの中のデータに対して直接マップ可能なインジケータである。それらは本発明のインストレーション時にセットアップされ、そしてエンド・ユーザが変更することはできない。「レポート（Reports）」または「InfoFrameTM」はテキストおよびグラフィック（例えば、グラフ、表）の形でデータベースからのデータを表示する複合ドキュメントである。レポートはInfoFrameTMの定義が実行された結果作成するものである。InfoFrameTMはHTMLフォーマットとすることができ、そしてOLE2．0に適合することができる。

【0031】「制限された属性（Restricted Attribute）」はそのセグメントを定義する時に制限されたセグメントの属性である。製品および価格は「高価なシャツ」の制限された属性であり得る。「セグメント（Segments）」は意味のある属性または複数の属性を共通に持っているコンセプトの内部で定義されるユーザ定義のグループである。例えば、セグメント「年配の顧客」はその年令が65才を超えている顧客とすることができる。

【0032】セグメントはパーティションの一部である。実際に、セグメントは1つまたはいくつかの属性についての制限によって定義されるデータのサブセットである。セグメントがいくつかの属性に定義されている場合、それはいくつかのパーティションの一部である可能性があり、制限された各属性に対して1つある（直ぐ後で示されるように、これは、孤立しているセグメントがあったとして、それがどのセグメントに属する「べきであるか」を判定することができず、したがって、その自然の親を階層の中で決定することができないことを意味する）。セグメントの集合はサブセットの関係の下に「セグメント階層」を形成し、そのルートはディメンションのメンバのすべてのを含んでいる「トップレベル・セグメント」である。

【0033】「構造化問合わせ言語（Structured Query Language（SQL））」は関係データベースの内容を見るための構造化された言語である。「要約化InfoFrameTM（Summarization InfoFrameTM）」は1つまたはそれ以上指定されたセグメントにわたる指定されたインジケータのロールアップ、すなわち、要約を示す1つのタイプのInfoFrameTMである。このレポートの中で特定のインジケータを選択することによって、指定された期間の間の「勝者」および「敗者」を示しているInfoFrameTMを自動的に作成することができる。

【0034】「システム・アドミニストレータ（System Administrators」（MDTアドミニストレータ、またはMDTA）はデータベースについて、および1つの組織のデータ構造についてよく知っているシステム10のユーザである。システム・アドミ

ニストレータの肩書きは「データベース・アドミニストレータ（ｄａｔａｂａｓｅ　ａｄｍｉｎｉｓｔｒａｔｏｒ）」（ＤＢＡ）であることが多い。「テキスト生成規則（Ｔｅｘｔ　Ｇｅｎｅｒａｔｉｏｎ　Ｒｕｌｅ）はいくつかのヒューリスティックなルールが満足されている時にＩｎｆｏＦｒａｍｅ™の中に挿入されなければならないテキストを指定する。

【００３５】「トレンド分析（Ｔｒｃｎｄ　Ａｎａｌｙｓｉｓ）ＩｎｆｏＦｒａｍｅ™」は、定義された時、指定の期間にわたる特定のインジケータまたは複数のインジケータに対する傾向を示す一種のＩｎｆｏＦｒａｍｅ™である。この分析は過去のアクティビティによるパターンを識別することによって、将来を予測するための支援となり得る。クライアントのサブシステム１２はグラフィカル・ユーザ・インターフェース（ＧＵＩ）４０を備えていて、ユーザがＩｎｆｏＦｒａｍｅ™に対して選択してパラメータを指定し、ＩｎｆｏＦｒａｍｅ™を表示し、ＩｎｆｏＦｒａｍｅ™をプリントし、そしてＩｎｆｏＦｒａｍｅ™を保存することができる単独のアプリケーション・プログラムである。最後に、ユーザは複合のインジケータおよびセグメントを定義し、アナリストを生成し、測定項目間の関係を定義し、あるいはアナリストのスケジュールを変更することができる。

【００３６】ＤＡＩサブシステム１４はグラフィカルなユーザ要求を変換し、システムのテンプレートを選択し、データ・ビューを操作し、そしてデータ・ウェアハウス２４からデータを検索するためのディメンション的な問合わせを生成するためのインテリジェントなミドルウェアを提供する。また、それはデフォールトのパラメータを選定するため、レイアウトおよびディスプレイのフォーマットを選定するため、およびデータからテキストを生成するためのルールも含んでいる。ＤＡＩサブシステム１４はユーザが選択したＩｎｆｏＦｒａｍｅ™をインスタンシエートするため、およびこのインスタンシエーションの中で使われるいくつかの種類のＭｅｔａｄａｔａ™２５を管理することを担当する。このＭｅｔａｄａｔａ™２５はデータ・ウェアハウス２４の中の関係データのカスタマイズ可能な「ディメンション化（ｄｉｍｅｎｓｉｏｎａｌｉｚａｔｉｏｎ）」を提供するコンセプトおよびインジケータを表す。また、ＤＡＩサブシステム１４はクライアント・サブシステム１２の中で作成するＭｅｔａｄａｔａ™２５に対する更新を処理する。それは主としてＤＳＭサブシステム１６へそれらを渡すことによって行なわれる。

【００３７】ＤＳＭサブシステム１６はデータ・ウェアハウス２４からスキーマを読み出し、データ・ビューを生成し、そしてその２つの間のマッピングを生成する。また、それはＤＡＩサブシステム１４から受信されたディメンション的な問合わせを、ＳＱＬおよびパッケージに変換するマッピングを行い、そしてその結果を返す。スケジューラ・サブシステム１８はスケジュールされた時刻において、あるいは規則的なスケジュールにおいて実行されるアナリスト、あるいはデータベースの中でトリガ条件を規則的にテストしなければならない例外アナリストを開始することを担当する。要求された時間間隔が発生すると、スケジューラが立ち上がり、ＤＡＩサブシステム１４からスケジュールされたＩｎｆｏＦｒａｍｅ™の要求のリストを要求する。これらのリストから、スケジューラ・サブシステム１８はその時点の時間間隔においてどれを実行するべきかを判定し、そしてそれらの要求をそれらがクライアント・サブシステム１２から送信されたかのようにＤＡＩサブシステム１４へ送る。

【００３８】このようにサブシステム１０は３階層のアーキテクチャとして実装されている。クライアント・コンピュータ３０はクライアント・サブシステム１２を実行する。クライアント・コンピュータ３０はＷｉｎｄｏｗｓ　ＮＴ™、またはＷｉｎｄｏｗｓ　９５™を稼動することが好ましいが、他のオペレーティング・システムも考えられる。クライアント・サブシステム１２（図６～１２）はこれらのオペレーティング・システムで使うのに適している。ディスプレイ２２および入力装置２１によって、ユーザはＧＵＩ４０を表示することができ、そしてアナリストを生成する際に使われるＭｅｔａｄａｔａ™２５の選択を入力することができる。入力装置２１はキーボード、マウス、あるいは他のポインティング・デバイスであってよい。プリンタ２３によってユーザはＩｎｆｏＦｒａｍｅ™をプリントすることができる。記憶媒体２６によってユーザはＩｎｆｏＦｒａｍｅ™またはアラート・メッセージを格納することができる。

【００３９】サーバ・コンピュータ３２はＤＡＩサブシステム１４、ＤＳＭサブシステム１６およびスケジューラ・サブシステム１８を実行する。これらの３つのサブシステムは組み合わさってデータ・ウェアハウス２４からの情報を使ってクライアント・サブシステム１２からのユーザ要求を満たす。サーバ・コンピュータ３２はマルチプロセッサ・コンピュータであること、そしてＵＮＩＸ™オペレーティング・システムまたはＷｉｎｄｏｗｓ　ＮＴ™を実行することが好ましいが、他のコンピュータおよびオペレーティング・システム構成も本発明によって考えられる。

【００４０】クライアントおよびサーバのコンピュータ３０および３２はレポート要求に対して非同期に結合されていることが好ましい。すべての他の要求は同期的に満足される。クライアントとサーバのコンピュータ３０および３２の間の通信は、伝送制御プロトコル（ｔｒａｎｓｍｉｓｓｉｏｎ　ｃｏｎｔｒｏｌ　ｐｒｏｔｏｃｅｌ／インターネット・プロトコル（ｉｎｔｅｒｎｅｔ　ｐ

ｒｏｔｏｃｏｌ）（ＴＣＰ／ＩＰ）を通じて行なわれるのが好ましいが、他の伝送プロトコルも本発明によって考えられる。

【００４１】データベース・コンピュータ３４はデータ・ウェアハウス２４を含んでいる１つまたはそれ以上の記憶媒体３６を含む。データベース・コンピュータ３４は大規模な並列プロセッサ・コンピュータであって、ＵＮＩＸオペレーティング・システムまたはＷｉｎｄｏｗｓ　ＮＴ™を実行することが好ましいが、他のコンピュータおよびオペレーティング・システムの構成も本発明によって考えられる。データ・ウェアハウス２４はデータ・ウェアハウス２４に対するオープン・データベース・コネクト（Ｏｐｅｎ　Ｄａｔａｂａｓｅ　Ｃｏｎｎｅｃｔ）（ＯＤＢＣ）インターフェースをサポートする任意のコンピュータで稼働するのに適している。サーバ・コンピュータ３２とデータベース・コンピュータ３４との間の通信はＯＤＢＣを経由するのが好ましいが、他のデータベース・インターフェースも本発明によって考慮されている。

【００４２】ここで図２を参照すると、クライアント・サブシステム１２はユーザがシステム１０を制御するアプリケーション・プログラムであり、Ｗｉｎｄｏｗｓ　ＮＴ™またはＷｉｎｄｏｗｓ　９５™のオペレーティング・システムのトップで実行されるのに適している。クライアント・サブシステム１２はログイン・モジュール５０、フォルダ管理サブシステム５４、セグメント・ビルダ５５Ａ、測定項目ビルダ５５Ｂおよび測定項目間の関係ビルダ５５Ｃ、アナリスト定義サブシステム５６、サブシステム５３を表示するＩｎｆｏＦｒａｍｅ™およびＭＤＴアドミニストレータ用インターフェース５７を含む。

【００４３】ログイン・モジュール５０はクライアント・サブシステム１２の１つのコピーだけがコンピュータ３０上で稼働しているかどうかをチェックし、コンピュータ３０のローカル化をチェックし、コンピュータ３２に対して接続し、そしてユーザと対話してクライアント・サブシステム１２上にユーザをログオンさせる。ログオン時に、ログイン・モジュール５０はユーザの名前およびパスワードをチェックし、以前に定義されていた可能性のあるユーザ・プリファレンスを呼び出す。ログイン・モジュール５０によって扱われるユーザからの唯一つの要求はクライアント・サブシステム１２にログ・オンするための要求である。ログ・イン・モジュール５０は次の要求を発行する。

【００４４】

・単独プログラム実行	オペレーティング・システム（DOS™ NT™、Windows　95™など）に対して
・ローカライゼーションの呼出し	オペレーティング・システム（DOS™ NT™、Windows　95™など）に対して
・サーバに対して接続する	クライアント／サーバ・モジュールに対して
・サーバから切り離す	クライアント／サーバ・モジュールに対して
・ユーザを認証する	Metadata™のAPI60に対して
・メイン・メニューを実行する	メイン・メニュー51に対して
・管理メニューを実行する	MDTアドミニストレータ用インターフェース57に対して

【0045】ユーザがシステム・アドミニストレータであった場合、ログイン・モジュール50はMDTアドミニストレータ用インターフェース57の「セットアップ」メニュー項目を表示する。ユーザがエンド・ユーザであるか、あるいは知識作業者であった場合、メイン・メニューおよびツールバー・インターフェース51がサブシステム53～55に関係付けられているインターフェースとして表示される。MDTアドミニストレータ用インターフェース57はグローバルに利用できるユーザ定義のセグメントの作成およびコンセプトの生成と編集などの、システム管理タスクを実行するためにシステム・アドミニストレータによって使われる。インターフェース62はシステム・インストレーション時にシステム・アドミニストレータに対してのみ提供されることが好ましい。

【0046】フォルダ管理システム54はフォルダの階層を操作し、格納し、そしてそれらのフォルダの中に格納されるInfoFrames™およびエージェントに関連したすべての機能を扱う。また、クライアント・サブシステム12のスタートアップ時、およびそれ以降定期的に、新しく完成したInfoFrame™に対するDAIサブシステム14からの問合わせを処理する。また、フォルダ管理サブシステム54は次のものについての動作に対するユーザ要求を処理する。

【0047】・フォルダ（新規作成、削除、名称変更）・エージェント（編集、削除、即時実行、プリント）・InfoFrame™（ビュー、削除、アノテート、プリント［InfoFrame™のビュー・ウィ

ンドウと協調して］）
各フォルダは1つのフォルダ・オブジェクトによって表される。フォルダはチャイルド・フォルダのリスト、ＩｎｆｏＦｒａｍｅ™のリスト、およびエージェントのリストを格納する。フォルダ・オブジェクトはユーザ要求に応答してフォルダ管理サブシステム５４によって生成／削除される。
【００４８】サブシステム５５Ｂは新しい測定項目の生成、測定項目の更新、または既存の測定項目の削除を行う機能をユーザに提供する。この情報はＭｅｔａｄａｔａ™のＡＰＩ６０へ送られ、そしてその後、ユーザのＭｅｔａｄａｔａ™２５を更新するためにＤＡＩサブシステム１４へ送られる。サブシステム５５Ａはユーザに新しいセグメントの生成、セグメントの更新または既存のセグメントの削除を行う機能を提供する。この情報はＭｅｔａｄａｔａ™のＡＰＩ６０へ送られ、その後、そのユーザのＭｅｔａｄａｔａ™２５を更新するためにＤＡＩサブシステム１４へ送られる。
【００４９】最後に、サブシステム５５Ｃはユーザに測定項目間の関係の変更、および測定項目間の関係の制約を行うためのインターフェースを提供する。ユーザは現在の測定項目を選択し、それが増加または減少する時にその測定項目間の関係を評価するかどうかを選択する。ユーザは他の測定項目のリストから選択することができ、そして現在の測定項目に対するそれぞれの関係を定義することができる。これらの関係は「減少する」、「増加する」、または「現在の測定項目には関連付けられていない」の形式である。また、2つの測定項目間のすべての関係に制約を付けることができる。測定項目とその測定項目に課せられた制約との関係がＩｎｆｏＦｒａｍｅｓ™の中で使うためにコンピュータ３２上に保存される。
【００５０】アナリスト定義サブシステム５６は特定のレポートを作成するために必要なパラメータのユーザ選択に関連したすべての機能を処理する。また、ユーザがスケジュールされたレポートに対してアラートを定義し、そしてスケジュールできるようにする。ユーザは既存のアナリストの呼出し、フォルダ管理サブシステム５４の内部からのアナリストの削除、または新しいアナリストの生成を行うことができる。次の５種類のアナリストがある。
【００５１】・要約化
・セグメントの比較
・測定項目の比較
・変動分析
・トレンド分析
要約化のアナリストは次のユーザ選択の要求を必要とする。
・アナリスト名
・一次測定項目、他のオプションの測定項目
・一次セグメント、他のセグメント
・時間間隔
・オプションのスケジュール
・オプションのトリガ
・使用される年のタイプ
・オプションのトリガ・イベント（アラート・メッセージ、ＩｎｆｏＦｒａｍｅ™、別のアナリストの実行）
【００５２】セグメント比較アナリストは次のユーザ選択要求を必要とする。
・アナリスト名
・一次測定項目、
・一次セグメント、比較セグメント
・時間間隔
・オプションのスケジュール
・オプションのトリガ
・使用される年のタイプ
・オプションのトリガ・イベント（アラート・メッセージ、ＩｎｆｏＦｒａｍｅ™、別のアナリストの実行）
【００５３】測定項目評価アナリストは次のユーザ選択要求を必要とする。
・アナリスト名
・一次測定項目、比較測定項目
・一次セグメント、他のオプションのセグメント
・基本期間、比較期間
・オプションのスケジュール
・オプションのトリガ
・使用される年のタイプ
・オプションのトリガ・イベント（アラート・メッセージ、ＩｎｆｏＦｒａｍｅ™、別のアナリストの実行）
【００５４】変動分析アナリストは次のユーザ選択要求を必要とする。
・アナリスト名
・一次測定項目
・一次セグメント、他のオプションのセグメント
・基本期間、比較期間
・オプションのスケジュール
・オプションのトリガ
・使用される年のタイプ
・オプションのトリガ・イベント（アラート・メッセージ、ＩｎｆｏＦｒａｍｅ™、別のアナリストの実行）
【００５５】トレンド分析アナリストは次のユーザ選択要求を必要とする。
・アナリスト名
・一次測定項目
・一次セグメント、他のオプションのセグメント
．期間、時間間隔
・オプションのスケジュール
・オプションのトリガ
・使用される年のタイプ
・オプションのトリガ・イベント（アラート・メッセー

ジ、InfoFrame™、別のアナリストの実行）
【００５６】ユーザはアナリスト定義を保存または実行することができる。ユーザは各コンセプトの内部から１つのセグメントを選定するように制限され、ただし、ターゲット・セグメントの場合は例外であり、その場合は１つのセグメントだけを選択することができ、そしてその選択されたセグメントの２つ以上のチャイルド・パーティションを選択することができる。ユーザは１つのアナリストをスケジュールするために選択することができ、あるいは既存のスケジュールを変更または削除することができる。スケジュールされていないアナリストはそのユーザが指令した時に実行される。スケジュールされているアナリストは後日または定期的に実行するためにサーバに対してサブミットされる。
【００５７】ユーザは例外アナリストを指定するためにそのアナリストに対するトリガ条件を指定することができる。サーバにサブミットされた時、それはそのトリガ条件をテストするために定期的に実行され、そしてそのトリガ条件が発生した時は常に、アラートまたはInfoFrames™を返す。アナリスト定義サブシステム５６はフォルダ管理サブシステム５４に対して次の要求を行う。
【００５８】

Save（保存）	ユーザが選択されたアナリストに対して適切なパラメータを選択したかどうかをチェックする。既存のアナリスト定義を保存するためにフォルダ管理サブシステム５４に対して要求を送る。
Save As 名前をつけて保存）	選択されたアナリストに対してユーザが適切なパラメータを選択したかどうかをチェックする。既存のアナリスト定義を保存するためにフォルダ管理サブシステム５４に対して要求を送る。
Submit サブミット）	選択されたアナリストに対してユーザが適切なパラメータを選択したかどうかをチェックする。レポート作成をサブミットするためにフォルダ管理サブシステム５４に対して要求を送る。

【００５９】また、アナリスト定義サブシステムは次の要求をMetadata™のAPI６０に対して行う。
【００６０】

Get all Measures （すべての測定項目を得る）	その要求はダイアログの初期化の時点でその必要があるたびに、Ｍｅｔａｄａｔａ™ ＡＰＩ６０に対して行なわれる。
Get all Concepts （すべてのコンセプトを得る）	その要求はダイアログの初期化の時点でその必要があるたびに、Ｍｅｔａｄａｔａ™ ＡＰＩ６０サブシステムに対して行なわれる。
Get a Concept's Partitions （コンセプトの パーティションを得る）	この要求はユーザのコンセプトの選択に依存して行なわれる。
Get Partitions パーティションを得る）	この要求は定義されたセグメントのユーザ選択に依存して行なわれる。
Get Segments セグメントを得る）	この要求はパーティションのユーザ選択に依存して行なわれる。

【００６１】ＩｎｆｏＦｒａｍｅ™ビューイング・サブシステム５３はＷＹＳＩＷＹＧブラウザを含み、それはＩｎｆｏＦｒａｍｅ™５３がＩｎｆｏＦｒａｍｅ™を見るためにフォルダ管理サブシステム５４から通知を受けた時に、選択されたＩｎｆｏＦｒａｍｅ™を画面に表示する。ユーザが現在のＩｎｆｏＦｒａｍｅ™からドリル・ダウンすることを決定した場合、ＩｎｆｏＦｒａｍｅ™ビューイング・サブシステム５３は新しいレポート要求を送るためにフォルダ管理サブシステム５４へ知らせる。

【００６２】ユーザがＩｎｆｏＦｒａｍｅ™上でダブルクリックするか、あるいはファイル・メニュー・フォルダから「ｍｅｎｕ　ｉｔｅｍ－Ｖｉｅｗ（メニュー・アイテムの表示）」を選択した場合、フォルダ管理サブシステム５４はそのＩｎｆｏＦｒａｍｅ™を見るためにＩｎｆｏＦｒａｍｅ™ビューイング・サブシステムへ通知する。ユーザが現在のＩｎｆｏＦｒａｍｅ™からドリル・ダウンするためにハイパーテキスト上でクリックした時、ＩｎｆｏＦｒａｍｅ™ビューイング・サブシステム５３はそのドリル・ダウン情報をホルダー管理サブシステム５４に渡して、ＤＡＩサブシステム１４に対する新しいレポート要求を送る。

【００６３】ＩｎｆｏＦｒａｍｅ™ビューイング・サブシステム５３はパーサを含み、そのパーサはＩｎｆｏＦｒａｍｅ™をパースして、ＨＴＭＬで書かれている完全なレポートを抽出する。ＨＴＭＬファイルの中で、ＨＴＭＬタグは他のドキュメントまたはリソースに対してリンクしているドキュメント要素、構造、フォーマッティング、およびハイパーテキストを示す。次に、パーサは表示のためのすべての情報を出力する。本発明においては、そのハイパーリンクは新しいアナリストおよびＩｎｆｏＦｒａｍｅ™であってよい。

【００６４】ＩｎｆｏＦｒａｍｅ™ビューイング・サブシステム５３によって、ユーザはパーサによって収集された情報に基づいてディスプレイ２２によってテキスト、表、およびグラフを表示およびフォーマットすることができる。ユーザはその表示されたＩｎｆｏＦｒａｍｅ™を保存することができる。また、ユーザは１つのＩｎｆｏＦｒａｍｅ™をＵＮＩＣＯＤＥまたはＡＳＣＩＩコードのフォーマットでＨＴＭＬファイルとして保存することもできる。保存されたＨＴＭＬＩｎｆｏＦｒａｍｅ™はｅメールに付加してメールで送り出すことができる。バージョン３．０のＨＴＭＬブラウザ、またはそれと等価な任意のブラウザがＨＴＭＬのＩｎｆｏＦｒａｍｅ™を読むことができる。

【００６５】Ｍｅｔａｄａｔａ™のＡＰＩ６０はクライアント・サブシステム１２とＤＡＩサブシステム１４との間のほとんどの通信を処理する。これらの通信は４種類の基本データ、すなわち、Ｍｅｔａｄａｔａ™２５、ＩｎｆｏＦｒａｍｅ™、ユーザ・プロファイル、

およびデータ・ウェアハウスのスキーマに関係する。Ｍｅｔａｄａｔａ™の通信の場合、Ｍｅｔａｄａｔａ™のＡＰＩ６０はＭｅｔａｄａｔａ™２５を追加、削除および更新する機能を提供する。ＩｎｆｏＦｒａｍｅ™の場合、Ｍｅｔａｄａｔａ™のＡＰ１６０はレポートのステータスを獲得し、レポートを検索し、そしてレポート要求を取り消すための機能を提供する。ユーザ・プロファイルの場合、Ｍｅｔａｄａｔａ™のＡＰＩ６０はユーザの追加、ユーザの認証およびユーザの削除機能を提供する。データ・ウェアハウス・スキーマのための通信はそれを検索することである。

【００６６】ＭｅｔａｄａｔａのＡＰＩ６０によって、ユーザは操作を眺める新しい方法を定義することができる。ユーザはパブリック・セグメント、基本測定項目またはパブリックな測定項目を変更することはできない。しかし、ユーザは新しいインジケータおよび新しいセグメントを生成することができる。ユーザとシステム・アドミニストレータの代表的な組織において、システム・アドミニストレータだけが基本の操作測定項目を生成または変更することができる。アドミニストレータおよび知識作業者はパブリックの複合測定項目、パブリック・セグメントおよびパブリック測定項目間の関係を生成、編集または削除することができる。Ｍｅｔａｄａｔａ™のＡＰＩ６０は他のクライアント・サブシステムからの次の要求を処理する。

【００６７】

Ｍｅｔａｄａｔａ™を更新する	サブシステム５５Ａ／５５Ｂ／５５Ｃから
レポートのステータスを得る	フォルダ管理サブシステム５４から
レポートをゼネレートする	フォルダ管理サブシステム５４から
レポートを検索する	フォルダ管理サブシステム５４から
スキーマを検索する	ＭＤＴアドミニストレータ用インターフェース５７から
スケジュールを更新する	アナリスト定義サブシステム５６から
レポートをキャンセルする	アナリスト定義サブシステム５６から
ユーザを認証する	ログイン・モジュール５０から
ユーザを追加する	ＭＤＴアドミニストレータ用インターフェース５７から
ユーザを削除する	ＭＤＴアドミニストレータ用インターフェース５７から
ユーザのパスワードを更新する	ＭＤＴアドミニストレータ用インターフェース５７から

【００６８】ＭｅｔａｄａｔａＴＭのＡＰＩ６０は次の要求を直接にＤＡＩサブシステム１４に対して送る
・コンピュータ３２から切り離す
・ＤＡＩサブシステム１４に対してデータを送信する
・ＤＡＩサブシステム１４からデータを受信する
ここで図３を参照すると、ＤＡＩサブシステム１４はリターン・エリア・マネージャ７０、ＩｎｆｏＦｒａｍｅＴＭゼネレータ７２、ＭｅｔａｄａｔａＴＭ要求モジュール７４、ＭｅｔａｄａｔａＴＭリポジトリ７６、およびＭｅｔａｄａｔａＴＭロードおよび更新モジュール７８を含む。
【００６９】ＭｅｔａｄａｔａＴＭリポジトリ７６はデータ・ウェアハウス２４の中のＭｅｔａｄａｔａＴＭ２５の表現を含む。このＭｅｔａｄａｔａＴＭ２５はシス

テム10のコアである。それはウェアハウス24の中の関係データについてのカスタマイズ可能なビューを提供し、そしてInfoFrames™の仕様に対する主要ボキャブラリである。Metadata™リポジトリ76はデータ・ウェアハウス24の中の永続的なMetadata™表現からDSMサブシステム16によってスタートアップ時に移植される。Metadata™のリポジトリ76の中には下記の4種類の基本的なMetadata™25がある。

【0070】・「コンセプト（Concepts）」：コンセプトはデータをそれに沿って眺めることができるディメンションを表す。各ディメンションはその底流のデータの上に階層を強制的に作り、そしてディメンションを組み合わせて「ドリル・ダウン」または「ドリル・アップ」操作をドライブすることができる。例えば、単純な小売り業のアプリケーションには2つのコンセプト、すなわち、市場および製品がある。市場の階層は販売地域から構成されており、各販売地域がいくつかの州から構成されており、各州は一組の店舗から構成されている。製品の階層は一組の部門（家庭用電子製品、男性用衣類、ハードウェア）から構成され、各部門は製品カテゴリー（シャツ、靴、スラックス）から構成され、そして各カテゴリーは個々の製造者の製品系列から構成されている。時間はすべてのアプリケーションにおいて重要なディメンションであり、そしてシステム10の中で表現される。ユーザは新しいコンセプトを追加することができる（下記参照）。これらはMetadata™リポジトリ76の中のMetadata™25のすべてと同様に、データ・ウェアハウス24に実際に問い合わせするために、関係の形式（すなわち、SQL）にマップされなければならない。マッピングはディメンション的なクエリを処理するプロセスの間にDSMサブシステム16によって行なわれる（下記参照）。

【0071】・「インジケータ（Indicators）」：インジケータは対象とするデータの重要な測定項目である。例えば、製品の数量、価格または現在のストックはすべてインジケータである。クエリの中で時間を使うことによって、インジケータの概念がさらにリファインされる。例えば、「数量における変化」は2つの期間の間に適用される。

・「アラート（Alerts）」：アラートは本質的にデータについてのテストであるが、それらはMetadata™の一部ではない。それらはMetadata™の条件でアナリストの中で規定される。例えば、ユーザは製品の利用できるストックが或るパーセンテージだけ落ちた場合、適切なInfoFrame™を生成することを指定することができる。また、ユーザはそのトリガ条件をチェックする頻度を指定することもできる。アラートのリストはDAIサブシステム14によって維持され、スケジューラ・サブシステム18によって実行される。また、このMetadata™25はDAIサブシステム14に対しても利用可能であり、そしてInfoFrame™情報を作成するために使われる。

【0072】・「測定項目間の関係（Measure Relationships）」：測定項目間の関係は偶発性の単純な表現である。例えば、「売上げの増加は利益の増加を意味する」。この種のMetadata™25はInfoFrame™に対するサポート情報を生成するため、あるいは代わりに、測定項目間の関係の集合に対して反対方向に進む傾向をユーザに対して警告するために使われる。Metadata™25は最初に顧客のサイトにおいて本発明のインストレーション時に生成される。Metadata™25の生成のプロセスは図7～図11の中でより詳細に示されている。Metadata™25の中に含まれるものはその産業によって変わり（或るMetadata™25は業界特有であり、そしてその業界におけるすべての企業によって使われる）、本発明の特定の顧客、そしてその顧客のデータ・ウェアハウス24の構造によって変わる。インストレーション時に、いくつかの業界固有のMetadata™25が使われ、いくつかの企業固有のMetadata™25が生成される可能性があり、そしてMetadata™25をデータ・ウェアハウス24に対してマップする必要があるマッピング情報が生成される。すべてのMetadata™25は、マッピング情報を含めて、一組の関係テーブルの中に格納される。これらの関係テーブルはデータ・ウェアハウス24の中でキープされ、そして本発明によってそのユーザに対するレポートを生成するために使われる。

【0073】Metadata™要求モジュール74はクライアント・サブシステム12又はDAIサブシステム14のいずれかからの、Metadata™25に対するすべての要求を処理する。クライアント・サブシステム12はエンド・ユーザに対して定義されるべきMetadata™25をDAIサブシステム14から要求する．InfoFrame™ゼネレータ72はユーザに対するInfoFrame™をインスタンシエートすることの一部として、ディメンション的なクエリーを生成するためにMetadata™25を要求する。Metadata™25に対する要求は、例えば、特定のコンセプトのすべてのサブコンセプトに対する要求である可能性がある．

【0074】また、Metadata™要求モジュール74はクライアント・サブシステム12からのMctadata™の更新も処理する．ユーザはそのデータをグループ化するための新しいディメンションを指定することによって、新しいセグメントを追加する．このディメンションはそのウェアハウスのデータの中の既存のデータ属性によってサポートされなければならない。例

えば、製品は定価および割引価格を含むことができる．ユーザはその割引価格と定価との間の差のパーセントを使って指定された「割引ファクター」と呼ばれる新しいディメンションを指定することができ、そしてそれを３つの新しいセグメント、すなわち、大幅値引き製品、小幅値引き製品、および非値引き製品を生成するために使う．これらの新しいセグメントをそれ以降のＩｎｆｏＦｒａｍｅ™要求の中で使うことができ、そして、ユーザによって示された場合、Ｍｅｔａｄａｔａ™のロードおよび更新モジュール７８によってデータ・ウェアハウス２４の中にそれらを書き戻すことによって継続性が保たれる．

【００７５】１つのサブシステムが別のサブシステムからの処理および結果を要求するとき、その１つのサブシステムから別のサブシステムに対して要求構造が渡される。要求構造は送られる要求のタイプに従って変わる。しかし、ほとんどの要求には、識別フィールド、所有者、名前および要求の記述などのいくつかの共通の属性がある。コンセプト更新要求がクライアント・サブシステム１２からＤＡＩサブシステム１４に対して送られる。それはシステム・アドミニストレータによってのみ発行されることが好ましい。コンセプト更新要求はＭｅｔａｄａｔａ™２５に対して新しいコンセプトを追加するための要求である。その要求のフォーマットは次の通りである。

【００７６】

ＢＣ＿ＩＤ：	このコンセプトをユニークに識別するＩＤ
ＢＣ＿ＮＡＭＥ：	このコンセプトの名前
ＢＣ＿ＤＥＳＣ：	このコンセプトの記述
ＭＡＰＰＩＮＧ：	データ・ウェアハウス・テーブルに対するこのコンセプトのマッピング

【００７７】インジケータ更新要求はクライアント・サブシステム１２からＤＡＩサブシステム１４に対して送られる。インジケータ更新要求はＭｅｔａｄａｔａ™２５に対して新しいインジケータを追加するための要求である。インジケータ更新要求は主としてプリミティブおよび複合の要求を含む。プリミティブの要求のフォーマットは次の通りである。

【００７８】

BI_ID：	このインジケータをユニークに識別するID
OWNER：	このインジケータを生成したユーザ
BI_NAME：	このインジケータの名前
BI_DESC：	このインジケータの記述
MAPPING：	データ・ウェアハウスのテーブルに対するこのインジケータのマッピング
ROLLUP_OP：	ロールアップ操作を実行するための演算子

【0079】複合要求のフォーマットは次の通りである。

【0080】

BI_ID：	このインジケータをユニークに識別するID
BI_NAME：	このインジケータの名前
BI_DESC：	このインジケータの記述
EXP：	このインジケータの機能を記述する式

【0081】因果関係インジケータ更新要求はクライアント・サブシステム12からDAIサブシステム14に対して送られる。因果関係インジケータ更新要求はMetadata™ 25に対して新しい因果関係インジケータを追加する。その要求のフォーマットは次の通りである。

【0082】

CI_ID：	この因果関係インジケータをユニークに識別するID
OWNER：	この因果関係インジケータを生成したユーザ
CI_NAME：	この因果関係インジケータの名前
CI_DESC：	この因果関係インジケータの記述
BI_ID1：	この因果関係の独立変数であるインジケータ
OP：	この因果関係に対する演算子
BI_ID2：	この因果関係の従属変数であるインジケータ
RANGE：	OPが＋／－の時、それが＋である範囲、それが－である範囲を表す。

【００８３】スキーマ要求はクライアント・サブシステム１２からＤＡＩサブシステム１４に対して送られ、そしてシステム・アドミニストレータによってのみ発行される。スキーマ要求はデータ・ウエアハウス２４からデータの基本スキーマを検索するための要求である。このタイプの要求はＤＡＩサブシステム１４に対する単純なフォーマットされていないメッセージだけである。セグメント更新要求はクライアント・サブシステム１２からＤＡＩサブシステム１４に対して送られる。セグメント更新要求はＭｅｔａｄａｔａ™ ２５に対して新しいセグメントを追加するための要求である。セグメント更新要求のフォーマットは次の通りである。

【００８４】

SEG_ID：	このセグメントをユニークに識別するID
OWNER：	このセグメントを生成したユーザ
SEG_NAME：	このセグメントの名前
SEG_DESC：	このセグメントの記述
SEG_LEVEL：	このセグメントのセグメント階層の中のレベル
BC_ID：	このセグメントに対するコンセプト
ATTR_ID：	このセグメントに対する属性
OP：	このセグメントに対する演算子
VALUE：	このセグメントに対する値

【0085】InfoFrame™要求はクライアント・サブシステムからDAIサブシステムへ送られる。このタイプの要求はユーザ指定のセクションに基づいて新しいInfoFrame™を生成することである。この要求のフォーマットは次の通りである。

【0086】

SE_ID：	このInfoFrame™をユニークに識別するID
OWNER：	このInfoFrame™を生成したユーザ
SR_NAME：	このInfoFrame™の名前
SE_DESC：	このInfoFrame™の記述
SR_TYPE：	4種類のInfoFrame™のうちの一つ
BC_ID：	このInfoFrame™に対するコンセプト
SEG_ID：	このInfoFrame™に対するセグメント
TIME：	このInfoFrame™に対する時間間隔

【0087】ディメンション的なクエリーはDAIサブシステム14からDSMサブシステム16へ送られる。ディメンション的なクエリーはデータ・ウエアハウス24からのデータに対する要求を公式化する。DSMサブ

システム１６はディメンション的クエリーをＳＱＬ文に変換する。ＤＡＩサブシステム１４はディメンション的なクエリーをＭｅｔａｄａｔａ™のセグメント定義またはパーティション定義のリスト、Ｍｅｔａｄａｔａ™２５の測定項目定義のリストおよび測定項目値テーブルとしてＤＳＭサブシステムに対して通信する。ＤＳＭサブシステム１６はこれらをＳＱＬクエリーに変換し、それらをデータ・ウェアハウス２４に対してサブミットする。ＤＳＭに対してデータ・ウェアハウスから戻された結果は測定項目値テーブルの中に入れてＤＡＩに対して戻される。

【００８８】クライアント・サブシステム１２はＤＡＩサブシステム１４に対して次の出力を作り出す。

・コンセプト更新要求
・インジケータ更新要求
・因果関係インジケータ更新要求
・スキーマ要求
・セグメント更新要求
・ＩｎｆｏＦｒａｍｅ™要求
・キャンセル要求

【００８９】ＤＡＩサブシステム１４はクライアント・サブシステム１２に対して次の出力を提供する。

・コンセプトの構造
・インジケータの構造
・因果関係インジケータの構造
・スキーマの構造
・セグメントの構造
・ＩｎｆｏＦｒａｍｅ™
・エラー／ステータス・コード

【００９０】ＤＡＩサブシステム１４はスケジューラ・サブシステム１８に対して次の出力を提供する。

・アナリストのスケジュール要求
・アナリストの削除要求

ＤＡＩサブシステム１４はＤＳＭサブシステム１６に対して次の出力を提供する。

・ディメンション的クエリー
・Ｍｅｔａｄａｔａ™検索要求
・スキーマ要求

【００９１】ＤＳＭサブシステム１６はＤＡＩサブシステム１４に対して次の出力を提供する。

・更新されたＭｅｔａｄａｔａ™
・データ・ウエアハウスからのデータ
・データベース・スキーマ

ＤＳＭサブシステム１６はデータ・ウェアハウス２４に対して次の出力を提供する。

・ＳＱＬ文

ＤＳＭサブシステム１６はデータ・ウエアハウス２４から次の入力を受け取る。

・Ｍｅｔａｄａｔａ™
・データベース・スキーマ
・ウェアハウス・データ

スキーマ１８はＤＡＩサブシステム１４に対して次の出力を提供する。

・アナリストの定義

【００９２】Ｍｅｔａｄａｔａ™のロードおよび更新モジュール７８はシステムのスタートアップ時にＭｅｔａｄａｔａ™のリポジトリー７６をデータ・ウェアハウス２４の中に格納されている永続的なＭｅｔａｄａｔａ™で埋める。さらに、ユーザが新しいコンセプトを規定し、そしてそれらを保存したいことを示すとき、Ｍｅｔａｄａｔａ™のロードおよび更新モジュール７８はそれらを将来の使用のためにデータ・ウェアハウス２４の中に書き戻す。

【００９３】ＩｎｆｏＦｒａｍｅ™ゼネレータ７２はＤＡＩサブシステム１４の主要目的を実現する。ＩｎｆｏＦｒａｍｅ™の定義を含んでいるユーザのアナリストがＤＡＩによって受信されたとき、レポートの作成が開始される。ＤＡＩの中に実装されているセットから適切なドリル・ダウン・ヒューリスティックおよびテキスト作成ルールを選択するためにアナリストのタイプが使われる。ドリル・ダウン・ヒューリスティックはレポートされなければならないターゲット・セグメントの自由属性のセグメント間にデータ関係があるかどうかを判定するために使われる。テキスト作成ルールはそのターゲット・セグメントのどの機能がレポートされるべきか、および兄弟セグメント、すなわち、そのターゲット・セグメントの制限された属性の中の他のセグメントに対してどの関係がレポートされるべきかを判定するために、テキスト作成ルールが使われる。テキスト作成ルールは適切な出力としてローカライズ可能なテキスト、グラフまたはテーブルを指定することができる。レポート作成プロセスの出力はポータブルな複合ドキュメントを作るために広く使われている標準であるハイパー・テキスト・マークアップ言語（ＨＴＭＬ）の形式でクライアント・サブシステム１２に対して返される、完全にインスタンシエートされたＩｎｆｏＦｒａｍｅ™である。

【００９４】ＩｎｆｏＦｒａｍｅ™ゼネレータ７２は次のいくつかの種類の知識を備えている。

・アブストラクト・クエリーをディメンション的なクエリーにマップする方法についての知識
・デフォールトの選択フォーム（ＩｎｆｏＦｒａｍｅ™要求の中でユーザによっては作られない選択項目）を作成するためにＭｅｔａｄａｔａ™２５を使う方法についての知識
・Ｍｅｔａｄａｔａ™２５とウェアハウスから戻されたデータの両方を使って両方のテキストのコンポーネントの選択をガイドするための方法についての知識
・異なるタイプのグラフィック表現の選択をガイドするために、Ｍｅｔａｄａｔａ™２５とウェアハウスから戻されたデータの両方を使う方法についての知識

【００９５】例えば、要約のＩｎｆｏＦｒａｍｅ™は式数としてコンセプト、インジケータ、および期間をとることができる。レポート作成モジュールはユーザ選択のパラメータ、例えば、コンセプト「製品」、コンセプト・セグメント「男性用シャツ」、インジケータ「数量」および期間「１９９４年１２月」を使ってディメンション的なクエリーを生成する。このディメンション的なクエリーはデータおよびスキーマの操作サブシステムに対して送られ、そのサブシステムはこのクエリーをＳＱＬに変換し、実際にそれを実行する。それは計算されたデータをＤＡＩサブシステム１４に対して返し、そこで他のアブストラクト・クエリーがパレットの中に実際の数値を埋め込むことができる。

【００９６】他のアブストラクト・クエリーには条件文が関連付けられている。前の例を作りあげるために、要約システム・テンプレートの別の部分によってグラフの生成を規定することがでてき、ターゲット・ビジネス・インジケータ（数量）がターゲット・ビジネス・コンセプト（シャツ）のセグメントの間にどのように配分されるかを示している。この場合、レポート・ゼネレータ７２はセグメントの集合（この例においては、シャツの製造者を規定するディメンション）を返すためのＭｅｔａｄａｔａ™要求を作る。すべての数量情報がシャツの各製造者に対して要求される。ここで、追加の情報がグラフの種類の選択においてレポート・ゼネレータ７２をガイドする。例えば、セグメント（この場合は製造者）の数が少ない（例えば、７以下）場合、パイ・グラフが適切であり、それ以外の場合は棒グラフが好ましい。セグメントの数が非常に多い場合、ボトムの２０％（そのインジケータ、この場合は数量に関しての）を集約し、その集約したものにグラフの中で「その他（Ｏｔｈｅｒ）」を付けて使う。

【００９７】リターン・エリア・マネージャ７０はクライアント・サブシステム１２に対する配送を待っているユーザによる肯定的な結果によって、ＩｎｆｏＦｒａｍｅ™およびアラート評価を追跡管理する。ユーザがシステム１０にログインしたとき、クライアント・サブシステム１２はＤＡＩサブシステム１４にそのリターン・エリアの中にそのユーザに対するすべてのデータを返すよう要求を発行する。リターン・エリア・マネージャ７０はサーバ・コンピュータ３２上のそのリターン・エリアから情報を検索し、それをＤＡＩサブシステム１４経由でクライアント・コンピュータ３０へ送り返す。

【００９８】ここで図４を参照すると、ＤＳＭサブシステム１６はＳＱＬゼネレータ８０およびＭｅｔａｄａｔａ™クエリー・モジュール８２を含んでいる。ＳＱＬゼネレータ８０はＤＡＩサブシステム１４から受信されたディメンション的なクエリーを、データ・ウェアハウス２４からデータを検索するために使われるＳＱＬ文に変換する。コンセプトからデータベースのエンティティーへのマッピングはＭｅｔａｄａｔａ™２５の中に格納され、ＳＱＬ文のフォーマッティングにおいて使われる。ＳＱＬゼネレータ８０はＩｎｆｏＦｒａｍｅ™を生成する際に使うために、ＤＡＩサブシステム１４に対して提供する。

【００９９】Ｍｅｔａｄａｔａ™クエリー・ゼネレータ８２はＤＡＩサブシステム１４がサブミットされたＭｅｔａｄａｔａ™２５に対する要求を処理する。システムのスタートアップ時にＤＡＩサブシステム１４は知識ベースを初期化するために、すべてのＭｅｔａｄａｔａ™２５を要求する。また、ユーザが自分のセグメントを変更するときは常にＭｅｔａｄａｔａ™クエリー・ゼネレータ８２が呼び出され、ＤＡＩサブシステム１４にＭｅｔａｄａｔａ™の更新要求を発行させる。

【０１００】ここで図５を参照すると、スケジューラ・サブシステム１８はアラートおよびレポートのスケジューラ９０を含んでいる。スケジューラは待ち行列に入っているスケジュールされたアナリストを定期的にテストし、実行の時期が来ているものをＤＡＩサブシステム１４に対してディスパッチする。スケジューラはサブミットされたすべての例外アナリストをＤＡＩサブシステム１４に対してディスパッチし、それらがトリガ条件をテストできるようにする。スケジュールおよびトリガの期間はＭＤＴアドミニストレータによって独立に構成することができる。スケジューラはディスパッチャ２５１３の方法によって、ＣＤＡＩ　１４Ｂに対してアナリストを渡す（図３４）。

【０１０１】ここで図６～図１９を参照すると、クライアント・サブシステム１２およびその動作が詳細に示されている。クライアント・サブシステム１２はクライアント・サブシステム１２が実行されるときに現れる一次オーバーレイ９８を含む。オーバーレイ９８は３つのディスプレイ・エリア１００～１０４を共通のフォルダ・ウィンドウ、プルダウン・メニュー１０６、およびボタン１１０～１２０の中に含む。フォルダのウィンドウは最大化されて（図６の中に示されているように）その境界をなくすこと、サイズを変更すること、あるいはアイコンとしてクライアント・サブシステム１２の中に最小化することができる。このフォルダのウィンドウを閉じることはできない。

【０１０２】ディスプレイ・エリア１００はフォルダのリストを含んでいる。それはＩｎｆｏＦｒａｍｅ™およびそれらを生成するアナリストを編成する際にクライアント・サブシステム１２によって使われるメタファーを表わす。フォルダは入力装置２１によってそれを強調表示させてから、それを選択することによって開かれる。リストの中の最初のフォルダはクライアント・サブシステム１２が実行されるときにデフォールトで開かれる。

【０１０３】ディスプレイ・エリア１０２は選択された

フォルダの中にあるＩｎｆｏＦｒａｍｅ™のリストを含む。ＩｎｆｏＦｒａｍｅ™はそれを強調表示させてから、それを入力装置２１によって選択して表示することができる。そのＩｎｆｏＦｒａｍｅ™を含んでいる分析ウィンドウ１３６が現れる。そのウィンドウのタイトル・バーは、実行されたあらかじめ選択されている分析のタイプを示す。例えば、図１９において、「変動（ｃｈａｎｇｅ）」分析は実行する分析のタイプとしてユーザによってあらかじめ選択されたものである。分析ウィンドウ１３６は最大化して（図１９に示されているように）その境界をなくすこと、サイズを変更すること、あるいはクライアント・サブシステム１２の中で１つのアイコンとして最小化することができる。分析ウィンドウ１３６はボタン１２２（図１９）を選択することによって、あるいはＷｉｎｄｏｗｓ　３．１™、Ｗｉｎｄｏｗｓ　９５™、および他のＷｉｎｄｏｗｓの動作環境のユーザにとって良く知られている方法によって閉じることができる。

【０１０４】ディスプレイ・エリア１０４は選択されたフォルダの中にあるアナリストのリストを含む。アナリストはＩｎｆｏＦｒａｍｅ™を生成する目的のためにあらかじめ選択されたデータについて実行される、あらかじめ選択された操作の擬人化である。アナリストはそれを強調表示させ、入力装置２１によってそれを選択することによって表示することができる。アナリスト・ビルダのウィンドウ１３０（図７～図１１）がそのアナリストの内部に保存されているあらかじめ選択された設定を含んで現れる。そしてディスプレイ・エリア１０２の中にリストされている対応しているＩｎｆｏＦｒａｍｅ™を生成するために使われる（ディスプレイ・エリア１０２の中にリストされているＩｎｆｏＦｒａｍｅ™はディスプレイ・エリア１０４の中にリストされているアナリストと同じ順序に配列されており、そしてそのタイトルは対応しているアナリストのタイトルと同じである）。アナリスト・ビルダのウィンドウ１３０は最大化したり、サイズ変更したり、あるいはアイコンとして最小化することはできない。それはＷｉｎｄｏｗｓ３．１™、Ｗｉｎｄｏｗｓ　９５™、および他のＷｉｎｄｏｗｓの動作環境のユーザにとって良く知られている方法によってのみ閉じることができる。

【０１０５】ボタン１１０～１２２（図６）はプルダウン・メニュー１０６の内部の一次操作コマンドを実装し、そしてポインティング・デバイスを使って活性化される。ボタン１１０はアナリスト・ビルダのウィンドウ１３０（図７～図１１）を呼び出す。ボタン１１２は情報セットアップ・ウィンドウ１３２（図１２）の中にセグメント・ディバイダを呼び出す。ボタン１１６はディスプレイ・エリア１００～１０４の内部の選択されたファイルまたはフォルダを削除する。ボタン１１８は新しいフォルダを生成する。ボタン１２０はディスプレイ・エリア１０２から選択されたＩｎｆｏＦｒａｍｅ™によって分析ウィンドウ１３６を呼び出す。ボタン１２２はクライアント・サブシステム１２を閉じる。ボタン１５０はプリント・ボタンであり、ボタン１５１によってユーザが測定項目を生成することができ、そしてボタン１５２によってユーザが測定項目間の関係を生成または編集することができる。

【０１０６】図７～図１１を参照すると、アナリスト・ビルダのウィンドウ１３０によってユーザは選択されたデータが分析される方法を定義することができる。アナリストの名前はアナリスト名（Ａｎａｌｙｓｔ　Ｎａｍｅ）のフィールドの下に付けられる。分析のタイプは「分析のタイプ（Ｔｙｐｅ　ｏｆ　Ａｎａｌｙｓｉｓ）」フィールドの中で選ばれる。その分析を実装するのに使われる主要な測定項目が「主要測定項目（Ｐｒｉｍａｒｙ　Ｍｅａｓｕｒｅ）」フィールドの中で選定される。レポートされるべきセグメントは定義されたセグメント（ＤｅｆｉｎｅＳｅｇｍｅｎｔｓ）のリストから選定される。最後に、そのＩｎｆｏＦｒａｍｅ™に対する期間が「考慮されるタイム・スライス（ＴｉｍｅＳｌｉｃｅＣｏｎｓｉｄｅｒｅｄ）」フィールドの中で定義される。ＩｎｆｏＦｒａｍｅ™は「即時レポート（Ｒｅｐｏｒｔ　Ｎｏｗ）」ボタンを選択することによってただちに生成することができるか、あるいは「スケジュール・アナリスト（Ｓｃｈｅｄｕｌｅ　Ａｎａｌｙｓｔ）」ボタンを選択することによってＩｎｆｏＦｒａｍｅ™のバッチの一部としてスケジュールすることができる。

【０１０７】図１２を参照すると、情報セットアップ・ウィンドウ１３２の中のセグメント・ディバイダによってセグメントが生成され、変更され、あるいは削除されるようにすることができる。セグメントの説明が「説明（Ｄｅｓｃｒｉｐｔｉｏｎ）」フィールドの中に現れる。ユーザによってボタン８０１が活性化された時、図１３のウィンドウ１３２が立ち上げられ、ユーザがセグメント定義を編集することができる。

【０１０８】図１４を参照すると、情報の測定項目を情報セットアップ・ウィンドウ１３２の測定項目ディバイダの中で生成および修正することができる。各メッセージに対する名前が「測定項目名（Ｍｅａｓｕｒｅ　Ｎａｍｅ）」フィールドの中に現れる。測定項目の定義は「定義（Ｄｅｆｉｎｉｔｉｏｎ）」フィールドの中に現れる。算術演算子、タイム・スライスの制約、セグメントの制約、および他の測定項目からの制約を定義フィールドの下の対応しているボタンを使ってその定義の中に挿入することができる。図１５および図１６を参照すると、ウィンドウ１３２は測定項目を選択するため、およびセグメントを選択するためにそれぞれ表示することができる。

【０１０９】図１７を参照すると、測定項目間の関係を

情報セットアップ・ウィンドウ１３２の測定項目間の関係ディバイダの中で定義して変更することができる。測定項目間の関係は「Ｉｆ－Ｔｈｅｎ」文の形で定義される。主要な測定項目およびそれが増加するか減少するかが測定項目フィールドの中で選択され、それは「Ｉｆ－Ｔｈｅｎ」文の「Ｉｆ」の部分を表わす。関係付けられていないフィールドの中の測定項目はＩｆ－Ｔｈｅｎ文の「Ｔｈｅｎ」の部分を形成するために減少フィールドまたは増加フィールドのいずれかへ移動させることができる。図１８を参照すると、測定項目間の関係をその図のウィンドウ１３２の手段によって制限することができる。

【０１１０】ＩｎｆｏＦｒａｍｅ™のバッチを自動生産のために個々にスケジュールすることができる。ＩｎｆｏＦｒａｍｅ™のスケジューリングは定期的なＩｎｆｏＦｒａｍｅ™を必要とするユーザにとって特に有用である。ＩｎｆｏＦｒａｍｅ™の時間間隔は「時間間隔（Ｔｉｍｅ　Ｉｎｔｅｒｖａｌ）」フィールドの中で選択することができ、それは毎日、毎週、毎月のレポートオプションを提供する。図１９を参照すると、ＩｎｆｏＦｒａｍｅ™の一例が分析ウィンドウ１３６の中に示されている。実行される分析のタイプはＩｎｆｏＦｒａｍｅ™の中で示されており、タイトル・バーの中で「変動分析（Ｃｈａｎｇｅ　Ａｎａｌｙｓｉｓ）」として示されている。セグメント（情報セットアップ・ウィンドウ１３２のセグメント・ディバイダの中で以前に定義された）は「４歳以上の年齢を格納する（Ｓｔｏｒｅ　Ａｇｅｓ　Ｇｒｅａｔｅｒ　Ｔｈａｎ　３　Ｙｅａｒｓ）」である。その測定項目（情報セットアップ・ウィンドウ１３２の測定項目ディバイダの中で以前に定義された）は「同じ店舗の売上高」である。そのタイム・スライス（アナリスト・ビルダ・ウィンドウ１３０の「考慮されるタイム・スライス」フィールドの中で以前に定義された）は「１９９５年度対昨年度」である。

【０１１１】ＩｎｆｏＦｒａｍｅ™は主要測定項目、同じ店舗の売上高の中で発生した変化の簡潔なステートメントを提供し、そして同じ店舗の売上高、再モデル化された店舗に関連した測定項目、および情報セットアップ・ウィンドウ１３２の測定項目間の関係ディバイダの中で以前に定義された測定項目の中で発生した変化についての簡潔なステートメントを提供する。次に、ＩｎｆｏＦｒａｍｅ™は主要測定項目、同じ店舗の売上高における変化に対してグラフを含んだ説明を含んでいる。ＩｎｆｏＦｒａｍｅ™はその測定項目の結果を表わしているテキスト・データまたはグラフィック・データに対するハイパーリンクを表わしている、測定項目に関連付けられたＨＴＭＬの複数のインスタンスを含むことができる。

【０１１２】ここで図２０を参照すると、クライアント・サブシステム１２を使ってＭｅｔａｄａｔａ™２５を生成するための方法が「開始（ＳＴＡＲＴ）」１４０から開始されて示されている。ステップ１４１においてユーザがコンセプトを指定する

ステップ１４２において、ユーザはそのコンセプトに対する１つまたはそれ以上の属性を指定する。

【０１１３】ステップ１４４において、クライアント・サブシステム１２はデータ２４の中のテーブルのカラムのリストをユーザに提供する。ステップ１４６において、ユーザはすべての属性を１つのカラムに対してマップする。ユーザはそのコンセプトおよび属性についてのテキストによる記述を提供することができる。ステップ１４８において、ユーザはデータ・ウエアハウス２４の内部のテーブルの中の１つのカラムに対して１つのインジケータを「マッピング」することによって、１つまたはそれ以上のインジケータを指定する。

【０１１４】ステップ１５０において、クライアント・サブシステム１２は同様にインジケータをマップする目的のために、カラムのリストをユーザに提供する。ステップ１５２において、ユーザはマップされたインジケータに対する「メソッドのまとめ」を選択する。それはそのインジケータに対する値が集計される方法を指定する。システムは次の集計メソッドをサポートする。

・加算
・平均
・最小値
・最大値
・カウント値
・最後に入った期間
・最初に入った期間

【０１１５】ステップ１５４において、ユーザは測定項目の単位を選択し、そのインジケータが通貨であるかどうかを指定する。ユーザはオプションとしてそのインジケータについての複数のフォーム、そのインジケータの値の変化を記述するための動詞（ｖｅｒｂ）、そのインジケータをレポートするための精度およびそのインジケータのテキストによる説明を指定することができる。ステップ１５６において、クライアント・サブシステム１２はインジケータのカラムを備えているテーブルを属性のカラムを備えているテーブルと結合させることができる。

【０１１６】ステップ１５８において、クライアント・サブシステム１２はユーザが追加のコンセプトを入力したいかどうかを判定する。入力したい場合はステップ１４２へ戻る。入力したくない場合はステップ１６０において終了する。前記の説明は本発明の概要である。以降のセクションではさらにセクションに分割して本発明をさらに詳しく説明する。

【０１１７】＜２．クライアント・サブシステム１２＞クライアント・サブシステム１２が以下にさらに詳しく説明される。図２１はクライアント・サブシステム１２

のさらに詳細のブロック図を示している。クライアント・サブシステム１２は３つのサブシステムを含む。それらはユーザ・インターフェース（ＵＩ）１４０１、マネージャ１４０２、およびサーバのＡＰＩ１４０３である。その名前が意味するように、ユーザ・インターフェースのサブシステム１４０１によってユーザ１４０５はクライアント１２と対話することができる。この詳細のレベルにおいては、ユーザ・インターフェース・サブシステム１４０１はマネージャ１４０２とＡＰＩ１４０３の両方のサービスを使うことがわかる。マネージャ１４０２もサーバのＡＰＩ１４０３からのサービスを使用する。サーバのＡＰＩサブシステム１４０３はすべてのクライアント１２がＤＡＩサブシステム１４と行う対話を要約する高レベルのＡＰＩを提供する。クライアント１２をＤＡＩサブシステム１４との間のすべての通信はクライアント・サーバ・モジュール（ＣＳＭ）１４０４を通じて送られる。このモジュールについては以下にさらに説明される。

【０１１８】図２２は図２１のブロック図をさらに詳しくしたレベルでの、クライアント・サブシステム１２のブロック図を示している。ユーザ・インターフェース・サブシステム１４０１はユーザ１４０５に対して見えるプログラムの部分をすべて含んでいる。このサブシステムは標準のＭＳ－Ｗｉｎｄｏｗｓスタイルのプログラムとして実装することができるので、そのインターフェースの内部のほとんどのユニットはウィンドウかあるいはダイアログ・ボックスのいずれかである。そのインターフェースの中の各ウィンドウまたはダイアログ・ボックスは１つのメイン・クラスを持ち、その挙動を以下に詳細に示されるように定義する。いくつかのウィンドウまたは対話クラスも他のユーティリティー・クラスを使用する。それは適宜以下で定義される。

【０１１９】クライアント・サブシステム１２の中のコントロールの「トップ・レベル」はアプリケーション・オブジェクト１５１１である。アプリケーション・オブジェクト１５１１はＭｉｃｒｏｓｏｆｔ　Ｆｏｕｎｄａｔｉｏｎ　Ｃｌａｓｓ（ＭＦＣ）ライブラリのスタートアップ・コードによって自動的に作られる。アプリケーション・オブジェクトは２つの主要な事項、すなわち、ログインの検証およびメイン・フレームのウィンドウ表示の機能を有する。複数ドキュメント・インターフェース（ＭＤＩ）のアプリケーションの中のフレーム・ウィンドウはメニュー、ツールバー、およびステータスバーを所有し、そして子ウィンドウのオブジェクトを生成する。

【０１２０】ユーザ・ログイン・プロセスは２つのステップから構成されている。それらはユーザからユーザ名およびパスワードを得るステップと、それらをサーバのＡＰＩサブシステム１４０３のＣｏｎｎｅｃｔ関数に対して送るステップである。サーバ３２に対して試みられた接続から、４つの可能な結果がある。

・ログイン成功
・ログイン失敗
・ログインの失敗が多すぎる
・サーバ３２から応答がない；ネットワーク・ダウン

【０１２１】ログインに失敗すると、ログイン・ダイアログが再表示され、そしてユーザ１４０５は自分の名前および／またはパスワードを入力し直すことができる。ある回数（クライアント１２でなく、サーバ３２によって決められた回数）だけ失敗した後、サーバ３２は「失敗が多すぎる」の結果を返し、クライアント１２のプログラムはこの結果についてユーザ１４０５へ通知してから脱出する。ネットワークまたはサーバがダウンした場合、クライアント１２は「オフ・ライン」モードで立ち上がり、それによってユーザ１４０５は保存されたＩｎｆｏＦｒａｍｅ™を見ることができるが、アナリストを生成または編集すること、あるいはＩｎｆｏＦｒａｍｅ™の作成要求を送ることはできない。

【０１２２】接続に成功したとき、アプリケーションはメイン・フレームのウィンドウを表示する。接続に成功した結果、ユーザ１４０５がアドミニストレータ（ＭＤＴＡ）の特権を有するかどうかの指示が追加して返され、そうであった場合、そのフレーム・ウィンドウに通知されて、特別のメニュー・アイテムがイネーブルされるようになる。アプリケーション・オブジェクト１５１１は他のサブシステムの次の要求を行うことができる。

【０１２３】

使用される機能	サブシステム
Connect（接続）	セッション・マネジメントＡＰＩ（サーバのＡＰＩサブシステム１４０３）
Disconnect（切り離し）	セッション・マネジメントＡＰＩ（サーバのＡＰＩサブシステム１４０３）
Display Manager Window ディスプレイ ・マネージャのウィンドウ）	マネージャ・ウィンドウ「ＵＩサブシステム１４０１」

【０１２４】アプリケーション・オブジェタト１５１１はｃｌｎｔ＿Ａｐｐクラスのインスタンスである。それはｃｌｎｔ＿ＵｓｅｒＬｏｇｉｎＤｌｇおよびｃｌｎｔ＿ＭａｉｎＦｒａｍｅのそれぞれについて１つのインスタンスを生成する。クラスｃｌｎｔ＿ＡｐｐはＭＦＣのクラスＣＷｉｎＡｐｐのサブクラスである。本発明はＣＷｉｎＡｐｐの標準の挙動のほとんどを継承するが、ＩｎｉｔＩｎｓｔａｎｃｅ機能にオーバーライドし、その中でユーザ・ログイン・プロセスを実行し、成功した場合、自分のメイン・ウィンドウ、ｃｌｎｔ＿ＭａｉｎＦｒａｍｅのインスタンスを作る。

【０１２５】ｃｌｎｔ＿ＭａｉｎＦｒａｍｅはＭＦＣのクラスＣＭＤＩＦｒａｍｅＷｎｄのサブクラスである。ツールバー、およびメニューを初期化するため、そして初期マネージャ・ウインドウのインスタンス１５１２を生成するために、ＯｎＣｒｅａｔｅ機能をオーバーライドする。ｃｌｎｔ＿ＭａｉｎＦｒａｍｅインスタンスはいくつかのメニューおよびツールバーの要求を扱い、一方、他の要求はアクティブであるどれかの子ウィンドウ（下記のように、４つのマネージャ・ウィンドウ１５１２のうちの１つまたはＩｎｆｏＦｒａｍｅ™のビューワ・ウィンドウ１５１７）によって処理される。ｃｌｎｔ＿ＭａｉｎＦｒａｍｅのインスタンスはどの子ウィンドウがアクティブであるかによって変わるメニュー・アイテムをイネーブル／ディスエーブルすることも担当する。

【０１２６】ユーザ・ログインのダイアログは、ＭＦＣのクラスＣＤｉａｌｏｇのサブクラスであるｃｌｎｔ＿ＵｓｅｒＬｏｇｉｎＤｌｇクラスのインスタンスによって制御される。ｃｌｎｔ＿ＵｓｅｒＬｏｇｉｎＤｌｇのインスタンスはユーザに名前とパスワードを入力するよう求めるダイアログを表示する。その名前およびパスワードの文字列はユーザが「ＯＫ」ボタンをクリックすると、呼び出している関数へ返される。

【０１２７】Ｔｏｏｌｂａｒは、ＭＦＣのＣＴｏｏｌＢａｒのサブクラスであるクラスｃｌｎｔ＿ＴｏｏｌＢａｒのインスタンスによって制御される。ｌｎｔ＿ＴｏｏｌＢａｒはＣＴｏｏｌＢａｒからのすべての機能を継承し、そしてドラッグ・アンド・ドロップに対するサポートを追加する。ＣＴｏｏｌＢａｒのインスタンスは1つのフォルダのドロップ（Ｔｒａｓｈボタンの上への）、１つまたはそれ以上のアナリスト（「ごみ箱（Ｔｒａｓｈ）」、「即時実行（ＲｕｎＮｏｗ）」、または「表示（Ｖｉｅｗ）」ボタンの上への）、および１つまたはそれ以上のＩｎｆｏＦｒａｍｅ™（「ごみ箱（Ｔｒａｓｈ）、「表示（Ｖｉｅｗ）、および「プリント（Ｐｒｉｎｔ）」ボタンの上への）を受け入れる。

【０１２８】情報定義１５１５はセグメント、測定項目、および測定項目間の関係の追加、変更、または削除に関連したすべての機能を含む。情報定義１５１５プロセスの中で３つのダイアログ・ボックスが使われる。それは編集される情報の各タイプごとに１つずつある。そのダイアログはｃｌｎｔＭａｉｎＦｒａｍｅによってインスタンシエートされる次のクラスのインスタンスによって制御される（メニューまたはツールバーによるユーザ要求に応答して）。

【０１２９】ｃｌｎｔ＿ＢｕｉｌｄＭｅａｓｕｒｅＤｌｇ。このダイアログによってユーザは既存の測定項目を更新または削除すること、あるいは新しい測定項目を生成することができる。

ｃｌｎｔ＿ＢｕｉｌｄＳｅｇｍｅｎｔＤｌｇ。このダイアログによってユーザは既存のセグメントを更新または削除するか、あるいは属性の制限を定義することによって新しいセグメントを生成することができる。

clnt_BuildRelationDlg。このダイアログによってユーザは既存のMeasureRelationを更新または削除するか、あるいは新しい関係を定義することができる。

測定項目間の関係のダイアログは他のサブシステムからの次のサービスを使用する。

【0130】

使用される機能	サブシステム
GetMeasureRelationship 測定項目間の関係を得る)	Metadata™のAPI［サーバのAPIサブシステム1403］
AddMeasureRelationship 測定項目間の関係を追加する)	Metadata™のAPI［サーバのAPIサブシステム1403］
DeleteMeasureRelationship 測定項目間の関係を削除する)	Metadata™のAPI［サーバのAPIサブシステム1403］
UpdateMeasureRelationship 測定項目間の関係を更新する)	Metadata™のAPI［サーバのAPIサブシステム1403］

【0131】測定項目ダイアログは次のサービスを使用する。

【0132】

使用される機能	サブシステム
AddMeasure 測定項目を追加する)	Metadata™のAPI［サーバのAPIサブシステム1403］
DeleteMeasure 測定項目を削除する)	Metadata™のAPI［サーバのAPIサブシステム1403］
UpdateMeasure 測定項目を更新する)	Metadata™のAPI［サーバのAPIサブシステム1403］
GetAllAnalysts すべてのアナリストを得る)	マネージャAPI［マネージャAPIサブシステム1402］

【0133】セグメント・ダイアログは次のサービスを使用する。

【0134】

使用される機能	サブシステム
AddSegment セグメントを追加する）	Metadata™のAPI［サーバのAPIサブシステム1403］
UpdateSegment セグメントを更新する）	Metadata™のAPI［サーバのAPIサブシステム1403］
DeleteSegment セグメントを削除する）	Metadata™のAPI［サーバのAPIサブシステム1403］
GetAllAnalysts すべてのアナリストを得る）	マネージャAPI［マネージャAPIサブシステム1402］

【0135】情報セットアップのセクション1515は次のクラス、すなわち、ｃｌｎｔＢｕｉｌｄＲｅｌａｔｉｏｎｓｈｉｐＤｌｇ、同じくｃｌｎｔ＿ＢｕｉｌｄＭｅａｓｕｒｅＤｌｇ、ｃｌｎｔ＿ＢｕｉｌｄＳｅｇｍｅｎｔＤｌｇおよびｃｌｎｔＢｕｉｌｄＲｅｓｔｒｉｃｔＤｌｇ（すべてＭＦＣのＣＤｉａｌｏｇのサブクラス）のインスタンスによって制御される。ユーザ１４０５がプライベート・セグメントまたは測定項目を変更することを選択した場合、ｃｌｎｔ＿ＭｅａｓｕｒｅＤｅｆＤｌｇおよびｃｌｎｔ＿ＳｅｇｍｅｎｔＤｅｆＤｌｇオブジェクトは既存のアナリストおよびＩｎｆｏＦｒａｍｅ™のリストの中を縦覧することを担当し、そしてそのセグメントまたは測定項目が見つかった場合、そのオブジェクトは次のアクションをとる。

【0136】削除の場合：

・その削除によっていくつかのアナリストが正しく実行されなくなることを示すメッセージがユーザ１４０５に対して表示される。ユーザ１４０５にはこの削除によって影響されるアナリストのリストが提示される。削除されたセグメントまたは測定項目についてアナリストが実行されると、エラー・メッセージが返される。

Ｍｏｄｉｆｙ（変更）の場合：

・最新のセグメント／測定項目定義が常に使われる。古い定義は置き換えられる。ユーザの変更要求はＭｅｔａｄａｔａ™の即時更新のためにＤＡＩへ転送される（サーバのＡＰＩサブシステムを通じて）。

【0137】アナリスト・ビルダのダイアログ・ボックス１５１３によってユーザ１４０５は特定のＩｎｆｏＦｒａｍｅ™を生成するのに必要なパラメータを選択する（下記参照）。また、それによってユーザ１４０５は１つのアナリストに対するスケジューリングおよび／またはトリガ条件の定義のオプションが可能となる。このことが起こるようにするために、メインのアナリスト・ダイアログはユーザ１４０５がサブダイアログ、すなわち、Ｍｅａｓｕｒｅ（測定項目）、Ｓｅｇｍｅｎｔｓ（セグメント）ＴｉｍｅＳｌｉｃｅ（タイムスライス）、Ｓｃｈｅｄｕｌｅ（スケジュール）、およびＴｒｉｇｇｅｒ（トリガ）を完了するよう催促する。ユーザ・インターフェース・サブシステム１４０１の他の部分（すなわち、メニュー、ツールバー、またはマネージャ・ウィンドウ）はアナリスト・ビルダのダイアログ１５１３を呼出す。それはビュー／編集のために既存のアナリスト・オブジェクトをそれに渡すか、あるいはＮＵＬＬパラメータを渡して新しいアナリストが生成されるべきであることを示すかのいずれかによって行われる。

【0138】ユーザ１４０５が新しいアナリストを「Ｓａｖｅ（保存）」または「ＳａｖｅＡｓ（名前をつけて保存）」を既存のアナリストの上で要求するとき、ｃｌｎｔ＿ＡｎａｌｙｓｔＳｈｅｅｔダイアログはｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔオブジェクトをインスタンシエートする。ｃｌｎｔ＿ＡｎａｌｙｓｔＳｈｅｅｔダイアログは次の要求を他のサブシステムに対し

て行う。

【0139】

使用される機能	サブシステム
NewAnalyst	マネージャＡＰＩ［マネージャ・サブシステム１４０２］
UpdateAnalyst	マネージャＡＰＩ［マネージャ・サブシステム１４０２］
RunAnalystNow	ＩｎｆｏＦｒａｍｅ™作成ＡＰＩ［サーバのＡＰＩサブシステム１４０３
SetFrameDefinition	ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔのＡＰＩ［マネージャ・サブシステム１４０２］
GetFrameDefinition	ＩｎｆｏＦｒａｍｅＲｅｑｕｃｓｔのＡＰＩ［マネージャ・サブシステム１４０２］

【0140】ｃｌｎｔ＿ＡｎａｌｙｓｔＳｈｅｅｔクラスは次のクラス、すなわち、ｃｌｎｔ＿ＡｎａｌｙｓｔＭｅａｓｕｒｅＰａｇｅ、ｃｌｎｔ＿ＡｎａｌｙｓｔＳｅｇｍｅｎｔＰａｇｅ、ｃｌｎｔ＿ＡｎａｌｙｓｔＴｉｍｅＳｌｉｃｅＰａｇｅ、ｃｌｎｔ＿ＡｎａｌｙｓｔＳｃｈｅｄｕｌｅＰａｇｅ、ｃｌｎｔ＿ＡｎａｌｙｓｔＴｒｉｇｇｅｒＰａｇｅ（すべてＭＦＣのＣＰｒｏｐｅｒｔｙＰａｇｅのサブクラス）のサブダイアログをインスタンシエートする。これらはユーザ１４０５が順番に導かれるメイン・ダイアログ・ボックスの中の５つのパネルに対応する。また、アナリスト・サブシステム１５１３はｃｌｎｔ＿ＭｅｔａＴｒｅｅクラスおよびｃｌｎｔ＿ＭｅａｓｕｒｅＭａｐクラスを使用する。それらはＭｅｔａｄａｔａ™のＡＰＩを通してＭｅｔａｄａｔｅテーブルに対するアクセスを提供する。

【0141】ｃｌｎｔ＿ＡｎａｌｙｓｔＳｈｅｅｔサブダイアログはユーザ１４０５のアナリスト・タイプの選択にしたがって適切なコントロールで動的に埋められる。このダイアログ・インターフェースからのユーザ入力はｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔオブジェクトの中に保存され、そして保存されるべきマネージャ・サブシステムへ返される（また、スケジュールが存在する場合は、スケジューリングに対してサブミットされる　－　スケジューラ・サブシステム１８のさらに詳細に関しては下記参照）。ｃｌｎｔ＿ＡｎａｌｙｓｔＭｅａｓｕｒｅＰａｇｅは次の要求を他のサブシステムに対して行う。

【0142】

使用される機能	サブシステム
GetName （名前を得る）	InfoFrameDefinitionのAPI ［マネージャ・サブシステム1402］
SetName （名前を設定する）	InfoFrameDefinitionのAPI ［マネージャ・サブシステム1402］
GetFrameType	InfoFrameDefinitionのAPI

（フレーム・タイプを得る）	[マネージャ・サブシステム1402]
SetFrameType （フレーム・タイプを設定する）	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
GetTargetMeasure （ターゲットの測定項目を得る）	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetTargetMeasure （ターゲットの測定項目を設定する）	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
GetComparisonMeasure （比較測定項目を得る）	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetComparisonMeasure （比較測定項目を設定する）	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
GetAdditionalMList （追加のMリストを得る）	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetAdditionalMList （追加のMリストを設定する）	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]

【0143】clnt_AnalystScgment　　【0144】
Pageは次の要求を他のサブシステムに対して行う。

使用される機能	サブシステム

GetTargetSegment (ターゲット・セグメントを得る)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetTargetSegment (ターゲット・セグメントを設定する)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
GetComparisonSegment (比較セグメントを得る)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetComparisonSegment (比較セグメントを設定する)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
GetAdditionalSList (追加のＳリストを得る)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetAdditionalSList (追加のＳリストを設定する)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
GetPartitionList (パーティション・リストを得る)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetPartitionList (パーティション・リストを設定する)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
GetParentPartition (親のパーティションを得る)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]
SetParentPartition (親のパーティションを設定する)	InfoFrameDefinitionのAPI [マネージャ・サブシステム1402]

【0145】clnt_AnalystTimeSli　　cePageは次の要求を他のサブシステムに対して行

う。

【0146】

使用される機能	サブシステム
GetPeriodType (期間のタイプを得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
SetPeriodType (期間のタイプを設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetAnalysisType (分析のタイプを得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
SetAnalysisType (分析のタイプを設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetYearType (年のタイプを得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
SetYearType (年のタイプを設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetTrendInterval (トレンドの時間間隔を得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]

SetTrendInterval (トレンドの時間間隔を設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetDuration (継続時間を得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
SetDuration (継続時間を設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetNumDuration (継続時間の数を得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
SetNumDuration (継続時間の数を設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetBasePeriod (基本期間を得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
SetBasePeriod (基本期間を設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetBaseThruPeriod (基本の全体の期間を得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
SetBaseThruPeriod (基本の全体の期間を設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetCompPeriod	InfoFrameTimeSliceのAPI

(計算期間を得る)	[マネージャ・サブシステム1402]
SetCompPeriod (計算期間を設定する)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
Operator= (演算子＝)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]
GetTimeSlice (タイム・スライスを得る)	InfoFrameTimeSliceのAPI [マネージャ・サブシステム1402]

【0147】clnt_AnalystSchedulePageは次の要求を他のサブシステムに対して行う。

【0148】

機能	サブシステム
GetNumInterval (時間間隔の数を得る)	InfoFrameScheduleのAPI [マネージャ・サブシステム1402]
SetNumInterval (時間間隔の数を設定する)	InfoFrameScheduleのAPI [マネージャ・サブシステム1402]
GetInterval (時間間隔を得る)	InfoFrameScheduleのAPI [マネージャ・サブシステム1402]
SetInterval (時間間隔を設定する)	InfoFrameScheduleのAPI [マネージャ・サブシステム1402]

GetStartDate （開始日付を得る）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
SetStartDate （開始日付を設定する）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
GetNumLimit （リミットの数を得る）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
SetNumLimit （リミットの数を設定する）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
GetLimit （リミットを得る）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
SetLimit （リミットを設定する）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
GetScheduleFlag （スケジュール・フラグを得る）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
SetScheduleFlag （スケジュール・フラグを設定する）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
GetTriggerFlag （トリガ・フラグを得る）	InfoFrameScheduleのAPI ［マネージャ・サブシステム1402］
SetTriggerFlag	InfoFrameScheduleのAPI

（トリガ・フラグを設定する）	[マネージャ・サブシステム1402]
Operator= （演算子＝）	InfoFrameScheduleのAPI [マネージャ・サブシステム1402]
SetSchedule （スケジュールを設定する）	InfoFrameScheduleのAPI [マネージャ・サブシステム1402]

【0149】clnt_AnalystTrigger　　【0150】
Pageは次の要求を他のサブシステムに対して行う。

機能	サブシステム
GetTriggerList （トリガ・リストを得る）	InfoFrameTriggerのAPI [マネージャサブシステム1402]
SetTriggerList （トリガ・リストを設定する）	InfoFrameTriggerのAPI [マネージャサブシステム1402]
GetMessageFlag （メッセージ・フラグを得る）	InfoFrameTriggerのAPI [マネージャサブシステム1402]
SetMessageFlag （メッセージ・フラグを設定する）	InfoFrameTriggerのAPI [マネージャサブシステム1402]
GetFrameFlag （フレーム・フラグを得る）	InfoFramcTriggerのAPI [マネージャサブシステム1402]

SetFrameFlag （フレーム・フラグを設定する）	フレーム作成のアクションの状態を更新する
GetAnalystList （アナリストのリストを得る）	InfoFrameTriggerのAPI ［マネージャサブシステム1402］
SetAnalystList （アナリストのリストを設定する）	InfoFrameTriggerのAPI ［マネージャサブシステム1402］
Operator= （演算子＝）	InfoFrameTriggerのAPI ［マネージャサブシステム1402］
GetMeasure （測定項目を得る）	TriggerのAPI ［マネージャ・サブシステム1402］
SetMeasure （測定項目を設定する）	TriggerのAPI ［マネージャ・サブシステム1402］
GetOperator （演算子を得る）	TriggerのAPI ［マネージャ・サブシステム1402］
SetOperator （演算子を設定する）	TriggerのAPI［マネージャ・サブシステム1402］
GetOperand1 （オペランド１を得る）	TriggerのAPI ［マネージャ・サブシステム1402］
GetOperand2	TriggerのAPI

（オペランド２を得る）	［マネージャ・サブシステム1402］
SetOperand1 （オペランド１を設定する）	TriggerのAPI ［マネージャ・サブシステム1402］
SetOperand2 （オペランド２を設定する）	TriggerのAPI ［マネージャ・サブシステム1402］
GetValue1 （値１を得る）	TriggerのAPI ［マネージャ・サブシステム1402］
GetBalue2 （値２を得る）	TriggerのAPI ［マネージャ・サブシステム1402］
SetValue1 （値１を設定する）	TriggerのAPI ［マネージャ・サブシステム1402］
SetValue2 値２を設定する）	TriggerのAPI ［マネージャ・サブシステム1402］
Operator= 演算子＝）	TriggerのAPI ［マネージャ・サブシステム1402］
SetTrigger（トリガを設定する）	InfoFrameRequestのAPI ［マネージャサブシステム1402］

【０１５１】次は各ＩｎｆｏＦｒａｍｅ™のタイプに対するユーザ入力条件のリストである。（Ｒ＝必要、Ｏ＝オプション）

ｃｌｎｔ＿ＭｅａｓｕｒｅＤｌｇ：

【０１５２】

分析のタイプ	ターゲットの測定項目	追加の測定項目	比較の測定項目
変更	R	O	
セグメントの比較	R	O	
測定項目の比較	R		R
要約	R	O	
トレンド	R	O	

【0153】clnt_SegmentDlg:　　【0154】

分析のタイプ	ターゲット・セグメント	追加のセグメント	比較セグメント
変更	R	O	
セグメントの比較	R	O	R
比較の測定項目	R	O	
要約	R	O	
トレンド	R	O	

【0155】clnt_TimeSliceDlg:　　【0156】

分析のタイプ	基本期間	比較期間	時間間隔
変更	R	R	
セグメントの比較	R		
比較の測定項目	R		
要約	R		
トレンド	R		R

【0157】InfoFrame™のビューワ・ウィンドウ1517は画面上にInfoFrame™を表示する（下記参照）。InfoFrame™のデータを表示すること以外に、ビューワ1517はユーザ1405に対して「ホット・スポット」を提示することによって「ドリル・ダウン」機能をサポートし、そしてホット・スポットが選択されたときに適切な要求を発生する。また、InfoFrame™のビューワはInfoFrame™をアノテートするための機能をユーザに提供する。InfoFrame™ビューワ1517が生成されると、それはInfoFrame™のファイルの名前およびそのInfoFrame™のオブジェクトに対するポインタを受け取る。このデータはパースされ（さらに、埋め込まれたデータからグラフを作成するなどの処理も行われ）、その後、表示される。

【0158】InfoFrame™のビューワ・モジュール1517の中のパーサの機能はSaveAs要求に対しても使われる。生のInfoFrame™データが標準のHTMLデータに変換され（すなわち、MDT固有のグラフ・データが標準フォーマットのグラフィック・イメージに変換される）、そしてASCIIまたはユニコード文字のいずれかでファイルに書き込まれる。InfoFrame™のビューワ・ウィンドウ1517はInfoFrame™のプリント機能もサポートする。この機能はMFCのCDocumentおよびCScrollViewクラスによって提供される機能に基づいて作られる。InfoFrame™のビューワ・サブシステム1517は次の要求を他のサブシステムに対して行う。

【0159】

使用される機能	サブシステム
UpDateInfoFrame™	ManagerのAPI［マネージャ・サブシステム1402］
DrillDown	ManagerのAPI［マネージャ・サブシステム1402］

【0160】パーサ（Perser）：clnt_Parserクラスは次の3つの機能によってクライアント1212に対してHTMLのパース機能を提供する。

【0161】

提供されるサービス	説明

doParse	ＩｎｆｏＦｒａｍｅ™のビューワによって呼出され、この機能は与えられたＨＴＭＬデータをパースし、そしてそれぞれがＨＴＭＬデータの１つの要素を表わしているｃｌｎｔ＿Ｔａｇオブジェクトのリストを返す。ｃｌｎｔ＿Ｔａｇオブジェクトはネスティングが保たれるように、サブタグのリストを含むことができる。
SaveASHTML_Unicode	ユーザ１４０５がＨＴＭＬファイルを保存するよう要求したときにマネージャ１４０２によって呼出される。ＨＴＭＬデータをパースして非標準のＨＴＭＬ要素を標準のデータで置き換える（たとえば、生のグラフ・データはグラフィック・イメージに変換されなければならない）。ユニコード文字を使って、変換されたデータをファイルに書き込む。
SaveAsHTML_ASCII	文字がＡＳＣＩＩとして書き出されること以外は上記に同じ。

【０１６２】ビューワ（Ｖｉｅｗｅｒ）：ビューワはＭＦＣのドキュメント／ビューのアーキテクチャーを使って実装される。クラスｃｌｎｔ＿ＶｉｅｗｅｒはＣＳｃｒｏｌｌＶｉｅｗ（ＭＦＣ）のサブクラスである。このクラスは自動スクロールを提供する。クラスｃｌｎｔ＿ＰａｒｓｅｒＤｏｃはＣＤｏｃｕｍｅｎｔのサブクラスである。生成時に、それはｃｌｎｔ＿ＰａｒｓｅｒオブジェクトをインスタンシエートしてそのＨＴＭＬデータをパースする。次に、ｃｌｎｔ＿Ｖｉｅｗｅｒはその返されたｃｌｎｔ＿Ｔａｇオブジェクトのリストを縦覧し、それぞれの視覚表現をウィンドウの中に置く。

【０１６３】次のコントロールの集合がユーザ・インターフェース・サブシステム１４０１によって使われる。ｃｌｎｔ＿ＴｒｅｅＣｔｒｌ　ＭＦＣのＣＴｒｅｅＣｔｒｌからではなく、このクラスからのセグメントおよび／パーティションの継承を表わしているすべてのダイアログ・コントロール。また、ｃｌｎｔ＿ＭｅｔａＴｒｅｅコントロールもこのクラスから継承する。

ｃｌｎｔ＿ＭｅｔａＴｒｅｅ　このコントロールは階層的なフォーマットでＭｅｔａｄａｔａ™のセグメントおよびパーティションを表わすために使われる。次のダイアログはこのコントロールのサブクラスである。：ｃｌｎｔ　ＡｎａｌｙｓｔＳｅｇｍｅｎｔＰａｇｅ、ｃｌｎｔ＿ＢｕｉｌｄＳｅｇｍｅｎｔＤｌｇ、ｃｌｎｔ＿ＲｅｓｔｒｉｃｔＭｅａｓｕｒｅＤｌｇ。

【０１６４】ｃｌｎｔ＿ＴｏｐＬｅｖｅｌＳｅｇｍｅｎｔＣｏｍｂｏ　このコントロールはドロップダウン・コンボボックスの中のすべてのＭｅｔａｄａｔａ™のトップ・レベルのセグメントを表わすために使われる。次のダイアログはこのコントロールのサブクラスである。：ｃｌｎｔ＿ＡｎａｌｙｓｔＳｅｇｍｅｎｔＰａｇｅ、ｃｌｎｔ＿ＢｕｉｌｄＳｅｇｍｅｎｔＤｌｇ、ｃｌｎｔ＿ＲｅｓｔｒｉｃｔＭｅａｓｕｒｅＤｌｇ。

ｃｌｎｔ＿ＤｕｒａｔｉｏｎＣｏｍｂｏ　このコントロールはドロップダウン・コンボボックスのフォーマットでのユーザの条件付き演算子の選択項目を表わす。次の

ダイアログはこのコントロールのサブクラスである。：clnt_AnalystPeriodPage、clnt_AnalystSchedulePage。

【0165】clnt_OperatorCombo このコントロールはドロップダウン・コンボボックスのフォーマットでのユーザの条件付き演算子の選択項目を表わす。次のダイアログはこのコントロールのサブクラスである。：clnt_AnalystPeriodPage、clnt_AnalystSchedulePage。

clnt_DateEdit　このコントロールはローカルの日付を表わすために使われる。それはユーザの入力をチェックし、その日付をそのローカルに適切な日付にフォーマットする。次のダイアログはこのコントロールのサブクラスである。：clnt_AnalystPeriodPage、clnt_AnalystSchedulePage。

clnt_ReadOnlyListBox　このコントロールは非選択のリストボックスのために使われる。他のサブクラスからの依存性はない。次のダイアログはこのこのコントロールのサブクラスである。：clnt_BuildSegmentDlg。

clnt_MetaTreeコントロールは他のサブシステムからの次のサービスを使用する。

【0166】

使用される機能	サブシステム
GetSegment （セグメントを得る）	Metadata™のAPI［サーバのAPIサブシステム 1403］
GetPartition （パーティションを得る）	Metadata™のAPI［サーバのAPIサブシステム 1403］

【0167】clnt_TopLebelSegmentComboは他のサブシステムからの次のサービスを使用する。

【0168】

使用される機能	サブシステム
GetSegment （セグメントを得る）	Metadata™のAPI［サーバのAPIサブシステム 1403］

【0169】clnt_MeasureComboは他のサブシステムからの次のサービスを使用する。

【0170】

使用される機能	サブシステム
GetBasicMeasure （基本測定項目を得る）	Metadata™のAPI［サーバのAPIサブシステム 1403］
GetCompositMeasure （複合測定項目を得る）	Metadata™のAPI［サーバのAPIサブシステム 1403］

【0171】アドミニストレータ用インターフェース1　　516は2つのタスク、すなわち、ユーザ・アカウント

・セットアップ、およびMetadata™ビルダから構成される。ユーザ・アカウント・セットアップのダイアログによって、MDTA（アドミニストレータ）はログイン名、パスワード、およびユーザ・タイプを含むユーザ・アカウントを生成して管理することができる。Metadata™ビルダによって、MDTAはディメンション、属性、および基本測定項目を定義すること、セグメントを生成すること、時刻値に対してカラムをマップすること、および年のタイプを定義することができる。ユーザ・アカウントの画面はサーバのAPIサブシステムからのclnt_UserLoginクラスを利用する。Metadata™ビルダの画面はサーバのAPIサブシステム1403によって提供されるほとんどすべてのMetadata™機能を利用する。これはclnt_Communications、Clnt_Dimension、clnt_Attribute、clnt_BasicMeasureおよびclnt_Segmentの各クラスのサービスを含む。また、それはデータ・ウェアハウスのスキーマにアクセスするためにclnt_Schemaクラスも使う。

【0172】ユーザ・アカウント・ダイアログはMFCのサブクラスである、clnt_UserAccountDlgのインスタンスによって制御される。clnt_UserAccountDlgがシステムの他の部分に対して提示するインターフェースは、CDialogオブジェクトに対する標準である。そのインターフェースが構築された後、DoModal（）が呼び出されてそのダイアログを表示する。DoModalに対する呼出しは、ユーザ1405が「キャンセル」または「クローズ」のボタンを押したときだけ戻る。

【0173】Metadata™ビルダのダイアログは「ウイザード」スタイルのダイアログである。これは、それが所定の順序での一連のサブダイアログを提示することを意味する。ユーザ1405は「次へ」および「戻る」のボタンを押してサブダイアログのリストを縦覧することができ、「キャンセル」を押してMetadata™ビルダから脱出することができる。そのウィザードの「フレーム」はMFCのCPropertySheetのサブクラスである、クラスclnt_MasterSetupによって実装される。clnt_MasterSetupのコンストラクタは各ダイアログ「ページ」（clnt_AttributeDefinition、clnt_AttributeMapping、clnt_AttributeValueDefinition、clnt_AutomaticSegments、clnt_BasicMeasureDefinition、clnt_BasicMeasurMap、clnt_DimensionDefinition、clnt_Joins、clnt_TimeDimension、clnt_YearDefinition）ごとに1つずつインスタンスを生成する。そのページはそれが表示される時に「ウイザード」の中にロードされる。これはMetadata™ビルダを単純にコンストラクトしてそのインスタンス上でDoModal（）を呼び出す他のクライアント・アプリケーション12に対してはトランスペアレントである。

【0174】各「ページ」はサーバのAPI1403のMetadata™クラスに対する呼出しを通して、その初期表示データをロードし、各ページはサーバのAPI1403を通してそのデータを更新することによって「保存」ボタンに対して応答する。clnt_MasterSetupは使用される各タイプのMetadata™に対する1つのリンク・リストを有している。各リストは0個以上のclnt_SetupObjectオブジェクトを含んでいる。clnt_SetupObjectオブジェクトは2つのデータ・メンバを含んでいる。1つはCObjectに対するポインタであり、もう1つはclnt_ObjectStateエニュメレーション（enumeration）である。clnt_Objectstateは4つの値、すなわち、STATE_EXISTING、STATE_NEW、STATE_DELETED、STATE_MODIFIEDのうちの1つを取ることができる。これらのリンク・リストはすべての「ウィザード」で利用できる。ユーザ1405がMetadata™オブジェクトを追加、削除、または変更するたびに、そのオブジェクトは適切なリンク・リストに追加される。これらのリンク・リストはどのオブジェクトをユーザ1405に対して表示するか、そしてどのオブジェクトをユーザ1405から隠すかを決定するために使われる。また、そのリンク・リストは「キャンセル」および「保存」のボタンによっても使われる。ユーザ1405が「キャンセル」ボタンを押すと、そのリンク・リストの中のすべてのオブジェクトが削除される。ユーザ1405が「保存」ボタンを押すと、そのリンク・リストの中のすべてのオブジェクトがアクセスされる。エニュメレーションの値がSTATE_EXISTINGであった場合、そのオブジェクトはそのリストから削除される。その値がSTATE_NEWであった場合、そのオブジェクトはサーバ上のMetadata™に対して追加され、そしてそのリストからは削除される。その値がSTATE_DELETEDであった場合、そのオブジェクトはサーバ上のMetadata™から削除され、そしてリストからも削除される。その値がSTATE_MODIFIEDであった場合、そのオブジェクトはサーバ上のMetadata™の中で更新され、そしてそのオブジェクトはリストから削除される。

【0175】「ウイザード」ページ上の「保存」ボタンはオブジェクトをある順序で追加、削除、および変更する。オブジェクトを削除する場合、次の表は削除される

べきオブジェクトを左のカラムに、そして存在していた場合にはクライアント１２上のリンク・リストから削除される関連付けられたオブジェクトを右のカラムに示している。左のカラムの中の行の順序はどのオブジェクト

オブジェクト　　関連付けられたオブジェクト

が最初に削除、追加、または変更されるかを定義する。ディメンションが最初に追加され、年の定義が最後に追加される。

【０１７６】

ディメンション	属性、セグメント
属性	列挙型属性値、制限された整数の属性値制限された浮動小数点数の属性値、セグメント、属性測定項目結合、属性属性結合
列挙型の属性値	＜なし＞
制限された整数の属性値	＜なし＞
制限された浮動小数点数の属性値	＜なし＞
セグメント	数値属性制限

数値属性制限	<なし>
列挙型文字列属性制限	<なし>
パーティション	<なし>
基本測定項目	属性と測定項目の結合
複合測定項目	定数、セグメント・リスト、属性と測定項目の結合
定数	<なし>
セグメント・リスト	<なし>
属性と測定項目の結合	<なし>
属性と属性の結合	<なし>
属性関係	測定項目間の関係範囲の制限、測定項目間の関係、大きさの制限、測定項目間の関係セグメントの制約
測定項目間の関係範囲の制限	<なし>
測定項目間の関係の大きさ	<なし>
測定項目間の関係セグメントの制約	<なし>

時間の定義	<なし>
時間のマッピング	<なし>
年の定義	<なし>

【0177】ユーザがサーバ34上のMetadata™ 25の中に既に存在するオブジェクトを削除すると、そのオブジェクトだけが削除される。そのオブジェクトに対する「関係付けられたオブジェクト」はDAIサブシステム14によって削除される。マネージャ・ウィンドウ1512はマネージャ・サブシステム1402によって記憶されているすべてのタイプのデータ（すなわち、ペンディングのInfoFrame™に関する情報以外に、フォルダ、アナリスト、およびInfoFrame™に対するアクセスを、ユーザ1405に提供する。

【0178】4種類のマネージャ・ウィンドウ1512があり、それぞれが次のように、このデータの異なるビューを提供している。

・アナリスト・リスト（すべてのアナリストのフラットなリスト）

・InfoFrame™のリスト（すべてのInfoFrame™のフラットなリスト）

・フォルダ・ビュー（フォルダの階層を含む；現在のフォルダの中のInfoFrame™およびアナリストを示す）

・ペンディングのキュー（DAI14の中で現在ペンディングになっているInfoFrame™のフラットなリスト）

【0179】ペンディングのキュー・ウィンドウは他の3つのマネージャ・ウィンドウと一緒に含まれていることに留意されたい。というのはコンストラクションおよびインターフェースの挙動が類似しているからである。それが表示するデータは実際には他の3つのウィンドウのデータとはかなり異っている。ドラッグ・アンド・ドロップの機能もマネージャ・ウィンドウ1512によってサポートされる。アナリストのリスト、InfoFrame™のリスト、およびフォルダ・ビューも「ドラッグ」操作のソースとなり得る（ユーザは1つのフォルダ、1つまたはそれ以上のアナリスト、あるいは1つまたはそれ以上のInfoFrame™をドラッグすることができる）。フォルダ・ビューは「ドラッグ」操作のデスティネーションともなり得る。

【0180】最初の3つのマネージャ・ウィンドウ・タイプ（アナリストのリスト、InfoFrame™のリスト、フォルダ・ビュー）は他のサブシステムからの次のサービスを使用する。

【0181】

使用される機能	サブシステム
GetRootFolder （ルートのフォルダを得る）	マネージャＡＰＩ［マネージャ・サブシステム１４０２］
GetTrashBin （ごみ箱のビンを得る）	マネージャＡＰＩ［マネージャ・サブシステム１４０２］
GetAllAnalyst （すべてのアナリストを得る）	マネージャＡＰＩ［マネージャ・サブシステム１４０２］
GetAllInfoFrames™ （すべてのInfoFrame™を得る）	マネージャＡＰＩ［マネージャ・サブシステム１４０２］
NewFolder	マネージャＡＰＩ［マネージャ・サブシ

（新しいフォルダ）	ステム１４０２]
RemoveFolder （フォルダを取り除く）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
MoveFolder （フォルダを移動する）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
SetFolderName （フォルダ名を設定する）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
MoveAnalyst （アナリストを移動する）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
RemoveAnalyst （アナリストを取り除く）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
MoveInfoFrame™ （ＩｎｆｏＦｒａｍｅ™ を移動する）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
RemoveInfoFrame™ （ＩｎｆｏＦｒａｍｅ™ を取り除く）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
EmptyTrash （ごみ箱を空にする）	マネージャＡＰＩ［マネージャ・サブシステム１４０２]
GetChildFolders	フォルダＡＰＩ［マネージャ・サブシス

（子のフォルダを得る）	テム１４０２]
GetInfoFrame™ （ＩｎｆｏＦｒａｍｅ™を得る）	フォルダＡＰＩ［マネージャ・サブシステム１４０２]
GetAnalysts （アナリストを得る）	フォルダＡＰＩ［マネージャ・サブシステム１４０２]
RemoveFolder （フォルダを取り除く）	フォルダＡＰＩ［マネージャ・サブシステム１４０２]
RunAnalystNow （アナリストをすぐに実行する）	ＩｎｆｏＦｒａｍｅ™作成ＡＰＩ［サーバのＡＰＩサブシステム１４０３]
ViewInfoFrame™ （ＩｎｆｏＦｒａｍｅ™を 表示する）	ＩｎｆｏＦｒａｍｅ™のビューワ・ウィンドウ［ユーザ・インターフェース・サブシステム１４０１]
run AnalystBuilder dialog （アナリスト・ビルダのダイアログを実行する）	アナリスト・ビルダ［ユーザ・インターフェース・サブシステム１４０１]

【０１８２】第4のマネージャ・ウィンドウ、「ペンディング・キュー」は他のサブシステムからの次のサービ　スを使用する。

【０１８３】

使用される機能	サブシステム
GetStatus （ステータスを得る）	ＩｎｆｏＦｒａｍｅ™作成ＡＰＩ［サーバのＡＰＩサブシステム１４０３]
CancelAnalyst （アナリストをキャンセルする）	ＩｎｆｏＦｒａｍｅ™作成ＡＰＩ［サーバのＡＰＩサブシステム１４０３]

【０１８４】４つの各マネージャ・ウィンドウ１５１２　　は1つの「フレーム」オブジェクトとその「フレーム」

の内部に置かれた1つまたはそれ以上の「コントロール」オブジェクトによって制御される。4つのすべてのケースにおいて、「フレーム」は1つだけのクラス、ｃｌｎｔ＿ＭａｎａｇｅｒＷｎｄ（ＭＦＣからのＣＭＤＩＣｈｉｌｄＷｎｄのサブクラス）によって表される。ｃｌｎｔ＿ＭａｎａｇｅｒＷｎｄオブジェクトはそれがどの「コントロール」オブジェクトを構築すべきかを示すために、インスタンシエーション時にパラメータ化される。そのサブクラスが示唆するように、それは標準のＭＤＩの子ウィンドウとして振る舞う。

【０１８５】フレーム・ウィンドウの中の「コントロール」オブジェクトはＭＦＣのクラスから継承し、それはまた、標準のＭＳ－Ｗｉｎｄｏｗｓのコントロールに対するラッパー（ｗｒａｐｐｅｒ）である。クラスｃｌｎｔ＿ＡｎａｌｙｓｔＣｔｒｌ、ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＣｔｒｌ、ｃｌｎｔ＿ＰｅｎｄｉｎｇＣｔｒｌはそれぞれＣＬｉｓｔＣｔｒｌから継承し、「カラム型の」リストの中のそれぞれのデータを表示する。クラスｃｌｎｔ＿ＦｏｌｄｅｒＣｔｒｌはＣＴｒｅｅＣｔｒｌから継承してＭＤＴフォルダのツリー状の階層を表示する。これらのクラスはそれが受け取る「ｓｔｙｌｅ（スタイル）」フラグに依存して、ｃｌｎｔ＿ＭａｎａｇｅｒＷｎｄによって、必要に応じて、インスタンシエートされる。すなわち、ｃｌｎｔ＿ＡｎａｌｙｓｔＣｔｒｌはＡＮＡＬＹＳＴＳモードおよびＦＯＬＤＥＲＳモードにおいて使われ、ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＣｔｒｌはＩＮＦＯＦＲＡＭＥＳおよびＦＯＬＤＥＲのモードにおいて使われ、ｃｌｎｔ＿ＦｏｌｄｅｒＣｔｒｌはＦＯＬＤＥＲＳモードにおいて、そしてｃｌｎｔ＿ＳａｖｅＡｓダイアログ・ボックス（アナリスト・ビルダの一部）によって使われ、ｃｌｎｔ＿ＰｅｎｄｉｎｇＣｔｒｌはＰＥＮＤＩＮＧモードにおいて使われる。ユーザ１４０５がドラッグ・アンド・ドロップ操作を開始すると、そのドラッグのソース・ウィンドウがｃｌｎｔ＿ＤｒａｇＷｎｄのインスタンスを構築する。その後、そのインスタンスがドラッグ・アンド・ドロップの残りを制御する。ｃｌｎｔ＿ＤｒａｇＷｎｄにはドラッグ中のオブジェクトまたはオブジェクトのリストに対するポインタおよび、ドラッグ中のオブジェクトのタイプを示す情報が与えられる。次に、それはカーソルが通過する任意のウィンドウに対して、そのウィンドウ上にそのオブジェクトをドロップしてもよいかどうかを尋ねるメッセージを送信する。ドロップをサポートするウィンドウはｃｌｎｔ＿ＦｏｌｄｅｒＣｔｒｌおよびｃｌｎｔ＿Ｔｏｏｌｂａｒである（セクション３．２．３参照）。ユーザ１４０５がマウス２１のボタンを解放すると、ｃｌｎｔ＿ＤｒａｇＷｎｄはドロップされるアイテムを受け入れるよう要求するメッセージをデスティネーション・ウィンドウに対して送信し、また、ドロップが完了したことを示すメッセージをソース・ウィンドウに対して送信する。

【０１８６】マネージャ・サブシステム１４０２はフォルダ１００の階層の操作、格納、および検索、そしてそれらのフォルダの中に格納されているＩｎｆｏＦｒａｍｅ™およびアナリストに関連したすべての機能を処理する。このデータを格納および検索することに関連したすべての機能がマネージャ・サブシステム１４０２の中にカプセル化されているので、本発明の代替実施形態においてフォルダ／ＩｎｆｏＦｒａｍｅ™／アナリストのデータ記憶がサーバ階層３２上に移動した場合でも、他のクライアント・サブシステム上でのインパクトは最小限で済む。

【０１８７】図２２から分かるように、マネージャ１４０２は4つのＡＰＩ、すなわち、マネージャ１５２１、フォルダ１５２２、アナリスト１５２３、およびＩｎｆｏＦｒａｍｅ™１５２４を提供する。これらのＡＰＩは次のセクションで説明される４つのクラスに対応する。マネージャ・サブシステム１５２１の中のメイン・クラスはｃｌｎｔ＿Ｍａｎａｇｅｒクラスである。３つのデータ・オブジェクト・クラス、すなわち、ｃｌｎｔ＿Ｆｏｌｄｅｒ、ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔ、およびｃｌｎｔ＿ＩｎｆｏＦｒａｍｅ™はｃｌｎｔ＿Ｍａｎａｇｅｒによって、そして他のサブシステムによって使われる。マネージャの機能に対するアクセスは普通は、フォルダ、アナリスト、またはＩｎｆｏＦｒａｍｅ™のリストを要求する、ｃｌｎｔ＿Ｍａｎａｇｅｒ自身に対する呼出しから開始される。これらのクエリーによって返されたオブジェクトは、次に、見るためおよび／または操作するためにユーザ１４０５に対して表示されるようにすることができる。そのデータ・オブジェクトのどれかに対する変更の要求はｃｌｎｔ＿Ｍａｎａｇｅｒを通過する。ｃｌｎｔ＿Ｍａｎａｇｅｒはディスク上へのその変更の格納を扱い、そして適用される場合はサーバのＡＰＩサブシステム１４０３に対するその変更の送信を処理する。

【０１８８】また、マネージャ・サブシステム１４０２は「ＴｒａｓｈＢｉｎ（ごみ箱）」の機能も提供する。すなわち、アナリストまたはＩｎｆｏＦｒａｍｅ™を削除する要求が受信されると、そのオブジェクトはごみ箱の中に置かれ、次のＥｍｐｔｙＴｒａｓｈ（ごみ箱を空にする）コマンドが受信されるまでは実際には削除されない。ＴｒａｓｈＢｉｎはクライアント１２のセッション間で継続性がある。ＴｒａｓｈＢｉｎはｃｌｎｔ＿Ｆｏｌｄｅｒクラスの1つのインスタンスとして実装されている。

【０１８９】クライアント・アプリケーションの中にはｃｌｎｔ＿Ｍａｎａｇｅｒクラスのインスタンス１５２１が正確に1つだけある。インスタンスが必ず1つだけ生成されるようにするために、そしてそれがグローバルに安全に利用できるようにするために、このクラスは

「シングルトン（Ｓｉｎｇｌｅｔｏｎ）」設計パターン（ガンマ（Ｇａｍｍａ）ほかによる「設計パターン：再使用可能なオブジェクト指向のソフトウェアの要素」Ａｄｄｉｓｎ－Ｗｅｓｌｅｙ、１９９５、ＩＳＢＮ０－２０１－６３３６１－２の中で説明されているような）を使用する。この設計パターンの中で、そのクラスは自分自身の１つのインスタンスに対するポインタを返すスタティック・メンバー関数を提供する。その関数は初めてそれが呼び出されたときに、そのインスタンスを自動的に生成する。そのクラスのコンストラクタはプロテクトされるので、そのクラスがどこか他の場所では決してインスタンシエートされないことが保証される。

【０１９０】ｃｌｎｔ＿Ｍａｎａｇｅｒクラスは他のサブシステムからの次の要求を処理する。

【０１９１】

提供されるサービス	説明
GetManager （マネージャを得る）	clnt_Managerの１つのインスタンスに対するポインタを返すスタティック・クラス関数。上記参照。
GetRootFolder （ルート・フォルダを得る）	トップのclnt_Folderオブジェクトに対するインタを返す。
GetTrashBin （ごみ箱を得る）	ごみ箱（実際には１つのclnt_Folderオブジクト）に対するポインタを返す。
GetAllAnalysis （すべてのアナリストを得る）	フォルダの階層に無関係にすべてのアナリストのリストを返す。
GetAllInfoFrame™ （すべてのInfoFrame™を得る）	フォルダの階層に無関係にすべてのＩｎｆｏＦｒａｍｅ™のリストを返す。
NewFolder	新しいフォルダを生成する。パラメータはそ

(新しいフォルダ)	の新しいフォルダに対する親を示す。新しく生成されたclnt_Folderオブジェクトに対すポインタを返す。
RemoveFolder (フォルダを取り除く)	与えられたフォルダ・オブジェクトを取り除く。そのサブフォルダ、アナリスト、およびInfoFrame™も取り除かれる。
MoveFolder (フォルダを移動する)	与えられたフォルダを新しい親フォルダへ移動する。
NewAnalyst (新しいアナリスト)	新しいclnt_InfoFrameRequestオブジェクト与えられたフォルダの中に格納し、スケジュールが存在している場合は、ScheduleAnalystサーバのAPIに対して送信する。
UpdateAnalyst (アナリストを更新する)	既存のclnt_InfoFrameRequestオブジェクト対する変更を格納し、スケジュールが存在している場合は、UpdateAnalystサーバのAPIに対して送信する。
MoveAnalyst (アナリストを移動する)	clnt_InfoFrameRequestオブジェクトを異なフォルダへ移動する。
RemoveAnalyst (アナリストを取り除く)	与えられたclnt_InfoFrameRequestオブジェトを削除する。スケジュールが存在している場合は、CancelAnalystをサーバのAPIサブシステムに対して送信する。

UpdateInfoFrame™ （InfoFrame™を更新する）	clnt_InfoFrame™オブジェクトに対する変更（普通は、たとえば、アノーテーションが追加されるか、あるいは生データがグラフに処理される時、そのHTMLデータに対する変更）を格納する。
MoveInfoFrame™ （InfoFrame™を移動する）	与えられたclnt_InfoFrame™オブジェクトを異なるフォルダへ移動する。
RemoveInfoFrame™ （InfoFrame™を取り除く）	与えられたclnt_InfoFrame™オブジェクトを削除する。
DrillDown （ドリル・ダウン）	要求されたドリル・ダウン・フレームが既に作成されていた場合、そのフレームを返す。それ以外の場合、フレーム作成要求をサーバのＡＰＩに対して送信する。
SaveInfoFrameAsMDTF （InfoFrame™をMDTファイルとして保存する）	ｅメールなどで送ることができるファイルを生成する。ファイルは「MDT InfoFrame™ファイル」である - クライアント・ソフトウェア１２を持っているものだけが使用できる。
SaveInfoFrameAsHTML （InfoFrame™をHTMLファイルとして保存する）	ｅメールなどで送ることができるファイルを生成する。ファイルは任意のHTMLブラウザ・プログラムによって見ることができる標準のHTMLファイル３．０のファイルである。この関数に対する１つのパラメータはASCIIまたUNICODEの出力がユーザによって要求されたどうかを示す。

ImportMDTFile （MDTファイルを インポートする）	SaveInfoFrameAsMDTFileコマンドによって以前に生成されたファイル読み込み、それをフォルダの中にInfoFrame™オブジェクトとて格納する。そのInfoFrame™は次に標準のカニズムを通して見ることができる。
EmptyTrash （ごみ箱を空にする）	現在ごみ箱のなかにあるすべてのアイテムを完全に削除する。

【0192】ｃｌｎｔ＿Ｍａｎａｇｅｒオブジェクトは他のサブシステムからの次のサービスを使用する。

【0193】

使用される機能	サブシステム
ScheduleAnalyst （アナリストを スケジュールする）	アナリストＡＰＩ［サーバのＡＰＩサブシステム１４０３］
UpdateAnalyst （アナリストを更新する）	アナリストＡＰＩ［サーバのＡＰＩサブシステム１４０３］
CancelAnalyst （アナリストを キャンセルする）	アナリストＡＰＩ［サーバのＡＰＩサブシステム１４０３］
GetStatus （ステータスを得る）	アナリストＡＰＩ［サーバのＡＰＩサブシステム１４０３］
RetrieveFrame （フレームを検索する）	ＩｎｆｏＦｒａｍｅ™作成ＡＰＩ［サーバのＡＰＩサブシステム１４０３］

【0194】クラスｃｌｎｔ＿Ｆｏｌｄｅｒのインスタンス１５２２はｃｌｎｔ＿Ｍａｎａｇｅｒオブジェクトによってのみインスタンシエートおよび削除される。他のサブシステムはｃｌｎｔ＿ＭａｎａｇｅｒのＧｅｔＲ

ｏｏｔＦｏｌｄｅｒ（）またはＧｅｔＴｒａｓｈＢｉｎ（）関数から開始してｃｌｎｔ＿Ｆｏｌｄｅｒインスタンスに対するアクセスを得る。ｃｌｎｔ＿Ｆｏｌｄｅｒオブジェクトは他のサブシステムからの次の要求を処理する。

【０１９５】

提供されるサービス	説明
GetFolderName（フォルダ名を得る）	このフォルダの名前を返す。
SetFolderName（フォルダ名を設定する）	このフォルダの名前を変更する。
GetChildFolders（子のフォルダを得る）	このフォルダの「子」であるｃｌｎｔ＿Ｆｏｌｄｅｒのリストを返す。
GetInfoFrame™（ＩｎｆｏＦｒａｍｅ™を得る）	このフォルダの中に格納されているｃｌｎｔ＿ＩｎｆｏＦｒａｍｅ™オブジェクトのリストを返す。
GetAnalysts（アナリストを得る）	このフォルダの中に格納されているclnt_InfoFrameRequestオブジェクトのリストをす。
RemoveFolder（フォルダを取り除く）	与えられたclnt_Folderオブジェクトを取除く。そのサブフォルダ、アナリスト、およびInfoFrame™も取り除かれる。

【０１９６】クラスｃｌｎｔ　ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔのインスタンス１５２３がｃｌｎｔ＿Ｍａｎａｇｅｒオブジェクトによって（保存されたアナリストをディスクから復元するとき）、そしてｃｌｎｔ＿ＡｎａｌｙｓｔＢｕｉｌｄｅｒダイアログ・クラスによって（新しいアナリストを生成する時または、現在のアナリストにおいてＳａｖｅＡｓを実行する時）生成される。他のサブシステムは普通はアナリスト・オブジェクトをそれぞれのフォルダ（ｃｌｎｔ＿Ｆｏｌｄｅｒ：：ＧｅｔＡｎａｌｙｓｔｓ（））から検索することによって、アナリスト・オブジェクトにアクセスする。ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔのインスタンスはｃｌｎｔ＿Ｍａｎａｇｅｒオブジェクトによってのみ削除される。ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔクラスは他のサブシステムからの次の要求を処理する。

【０１９７】

提供されるサービス	説明
GetName（名前を得る）	このアナリストの名前を返す。
SetName（名前を設定する）	このアナリストに対して新しい名前を割り当てる。

GetRequestID （要求ＩＤを得る）	マネージャによって割り当てられたユニークな要求ＩＤを返す。
SetRequestID （要求ＩＤを設定する）	そのアナリスト要求に対してユニークな要求ＩＤを割り当てる。
GetUserName （ユーザ名を得る）	ユーザの名前を返す。
SetUserName （ユーザ名を設定する）	このアナリスト要求に対してユーザ名を割り当てる。
GetFrameDef （フレームの定義を得る）	このアナリストに対してｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＤｅｆｉｎｉｔｉｏｎオブジェクトを返す。
SetFrameDef （フレームの定義を設定する）	アナリストのＦｒａｍｅＤｅｆｉｎｉｔｉｏｎオブジェクトを更新する。
GetSchedule （スケジュールを得る）	このアナリストに対するｃｌｎｔ＿Ｓｃｈｅｄｕｌｅオブジェクトを返す。
SetSchedule （スケジュールを設定する）	そのアナリストのスケジュール・オブジェクトを更新する。
GetTrigger（トリガを得る）	このアナリストに対するｃｌｎｔ＿Ｔｒｉｇｇｅｒオブジェクトを返す。

SetTrigger（トリガを得る）	そのアナリストのトリガ・オブジェクトを更新する。
GetContainingFolder（含んでいるフォルダを得る）	含んでいるオブジェクトに対するポインタを返す。
SetContainingFolder（含んでいるフォルダを設定する）	含んでいるフォルダ・オブジェクトに対するポインタを更新する。

【0198】clnt_InfoFrameRequestクラスは3つのヘルパー・クラスを通じてその仕事のほとんどを行う。clnt_FrameDefinitionクラスはこのアナリストが実行される（あるいはスケジュールされる）時に送信されるInfoFrame™作成要求の記述を格納する。clnt_FrameDefinitionクラスは他のサブシステムからの次の要求を処理する。

【0199】

提供されるサービス	説明
GetFolderID（フォルダのＩＤを得る）	clnt_Folderオブジェクトによってそアナリストに割り当てられたフォルダＩＤを返す。
SetFolderID（フォルダＩＤを設定する）	そのフォルダＩＤをそのアナリストに対して割り当てる。
GetAnalysisType（分析のタイプを得る）	この要求に対して選択された分析のタイプを返す。

SetAnalysisType （分析のタイプを設定する）	この要求に対して選択された分析のタイプを更新する。
GetTargetMeasure （ターゲット測定項目を得る）	この分析に対して選択されたターゲット測定項目を返す。
SetTargetMeasure （ターゲット測定項目を設定する）	この分析に対して選択されたターゲット測定項目を更新する。必要。
GetComparisonMeasure （比較測定項目を得る）	この分析に対して選択された比較測定項目を返す。測定項目比較分析に対してのみ必要。
SetComparisonMeasure （比較測定項目を設定する）	この分析に対して選択された比較測定項目を更新する。測定項目比較分析に対してのみ必要。
GetAdditionalMList （追加の測定項目リストを得る）	この分析に対して選択された追加の測定項目オブジェクトのリストを返す。オプション。
SetAdditionalMList （追加の測定項目リスト を設定する）	この分析に対して選択された追加の測定項目オブジェクトのリストを更新する。オプション。
GetTargetSegment （ターゲット・セグメントを得る）	この分析に対して選択されたターゲット・セグメントを返す。必要。

SetTargetSegment (ターゲット・セグメント を設定する)	この分析に対して選択されたターゲット・セグメントを更新する。必要。
GetComparisonSegment (比較セグメントを得る)	この分析に対して選択された比較セグメントを返す。セグメント比較分析に対してのみ必要。
SetComparisonSegment (比較セグメントを設定する)	この分析に対して選択された比較セグメントを更新する。セグメント比較分析に対してのみ必要。
GetAdditionalSList (追加のセグメント・オブジェクトのリストを得る)	この分析に対して選択された追加のセグメント・オブジェクトのリストを返す。オプション。
SetAdditionalSList (追加のセグメント・リストを設定する)	この分析に対して選択された追加のセグメント・オブジェクトのリストを更新する。オプション。
GetPartitionList (パーティションのリストを得る)	選択されたターゲット・パーティションのリストを返す。オプション。
SetPartitionList (パーティションのリスト を設定する)	選択されたターゲット・パーティションのリストを更新する。オプション。
GetParentPartition (親パーティションを得る)	選択されたターゲット・セグメントの親パーティションを返す。ターゲット

	・セグメントがトップ・レベルのセグメントでない場合にのみ必要。
SetParentPartition （親パーティションを設定する）	ターゲット・セグメントの親パーティションを更新する。ターゲット・セグメントがトップ・レベルのセグメントでない場合にのみ必要。
GetTimeSlice （タイム・スライスを得る）	この分析に対するタイム・スライス・オブジェクトに対するポインタを返す
SetTimeSlice （タイム・スライスを設定する）	この分析に対するタイム・スライス・オブジェクトに対するポインタを更新する。
Operator=（演算子＝）	そのオブジェクトを別のオブジェクトへコピーする。

【0200】clnt_InfoFrameTimeSliceクラスは他のサブシステムからの次の要求を処理する。

【0201】

提供されるサービス	説明
GetPeriodTypc （期間のタイプを得る）	その要求に対して選択されたタイム・スライスのタイプを返す。
SetPeriodType	その要求に対して選択されたタイム・

(期間のタイプを更新する)	スライスのタイプを更新する。
GetAnalysisType (分析のタイプを得る)	その要求に対して選択された分析のタイプを返す。
SetAnalysisType (分析のタイプを更新する)	その要求に対して選択された分析のタイプを更新する。
GetYearType (年の定義のタイプを得る)	その要求に対して選択された年の定義のタイプを返す。
SetYearType (年の定義のタイプを更新する)	その要求に対して選択された年の定義のタイプを更新する。
GetTrendInterval (トレンドの時間間隔を得る)	時間間隔を返す。トレンド分析に対してのみ必要。
SetTrendInterval (トレンドの時間間隔を設定する)	時間間隔を更新する。トレンド分析に対してのみ必要。
GetDuration (継続時間を得る)	継続時間を返す。
SetDuration (継続時間を設定する)	継続時間を更新する。
GetNumDuration (継続時間の数を得る)	継続時間の数を返す。

SetNumDuration （継続時間の数を設定する）	継続時間の数を更新する。
GetBasePeriod （基本の期間を得る）	特定の日付の基本期間を返す。
SetBasePeriod （基本の期間を設定する）	特定の日付の基本期間を更新する。
GetBaseThruPeriod （基本の全体期間を得る）	特定の日付の全体期間を返す。
SetBaseThruPeriod （基本の全体期間を設定する）	特定の日付の全体期間を更新する。
GetCompPeriod （比較期間を得る）	特定の日付の比較期間を返す。変動分析によってのみ必要。
SetCompPeriod （比較期間を設定する）	特定の日付の比較期間を更新する。変動分析によってのみ必要。
Operator=（演算子＝）	１つのタイム・スライス・オブジェクトを別のタイム・スライス・オブジェクトへコピーする。

【０２０２】ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＳｃｈｅｄｕｌｅクラスはそのアナリストに対するスケジュールの定義を格納する。ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＳｃｈｅｄｕｌｅクラスは他のサブシステムからの次の要求を処理する。

【０２０３】

提供されるサービス	説明
GetNumInterval （時間間隔の数を得る）	そのレポートが実行される必要がある時間間隔の数を返す。
SetNumInterval （時間間隔の数を設定する）	そのレポートが実行される必要がある時間間隔の数を更新する。
GetInterval （時間間隔を得る）	そのレポートが実行される必要がある継続時間を返す。
SetInterval （時間間隔を設定する）	そのレポートが実行される必要がある継続時間を更新する。
GetStartDate （開始日付を得る）	そのレポートが実行開始されるためにスケジュールされている日付を返す。
SetStartDate （開始日付を設定する）	そのレポートが実行開始されるためにスケジュールされている日付を更新する。
GetNumLimit （リミットの数を得る）	そのレポートが実行されるためにスケジュールされている期間の数を返す。
SetNumLimit （リミットの数を設定する）	そのレポートが実行開始されるためにスケジュールされている期間の数を更

	新する。
GetLimit （リミットを得る）	そのレポートが実行されるためにスケジュールされている回数に対する継続時間を返す。
SetLimit （リミットを設定する）	そのレポートが実行開始されるためにスケジュールされている回数に対する継続時間を更新する。
GetScheduleFlag （スケジュール・フラグを得る）	そのスケジュールのイネーブルまたはディスエーブルの状態を返す。
SetScheduleFlag （スケジュール・フラグ を設定する）	そのスケジュールのイネーブルまたはディスエーブルの状態を更新する。
GetTriggerFlag （トリガ・フラグを得る）	そのトリガのイネーブルまたはディスエーブルの状態を返す。
SetTriggerFlag （トリガ・フラグを設定する）	そのトリガのイネーブルまたはディスエーブルの状態を更新する。
Operator=（演算子＝）	１つのスケジュール・オブジェクトを別のスケジュール・オブジェクトへコピーする。

【0204】ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＴｒｉｇｇｅｒクラスはその分析が実行される前にチェックされるべき条件の定義を処理する。ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＴｒｉｇｇｅｒクラスは他のサブシステムからの次の要求を処理する。

【0205】

提供されるサービス	説明
GetTriggerList （トリガ・リストを得る）	そのアナリストによって定義されているトリガのリストを返す。
SetTriggerList （トリガ・リストを設定する）	そのアナリストによって定義されているトリガのリストを更新する。
GetMessageFlag （メッセージ・フラグを得る）	取られるべきアクションのユーザ選択によって変わるイネーブル／ディスエーブルの状態を返す。この場合、トリガが「真」になれば、メッセージが発生される。
SetMessageFlag （メッセージ・フラグを設定する）	メッセージ作成アクションの状態を更新する。
GetFrameFlag （フレーム・フラグを得る）	取られるべきアクションのユーザ選択によって変わるイネーブル／ディスエーブルの状態を返す。この場合、トリガが「真」になれば、フレームが作成される。
SetFrameFlag （フレーム・フラグを設定する）	フレーム作成アクションの状態を更新する。

GetAnalystList （アナリストのリストを得る）	トリガが「真」になった場合に作成されるべきアナリストのリストを返す。リストが空の場合、このアクションは選択されない。
SetAnalystList （アナリストのリストを設定する）	トリガが「真」になった場合に作成されるべきアナリストのリストを更新する。リストが空の場合、このアクションは選択されない。
Operator=（演算子＝）	そのオブジェクトを別のオブジェクトへコピーする。

【0206】クラスｃｌｎｔ＿ｔｒｉｇｇｅｒは単独のトリガ条件を含んでいる。ｃｌｎｔｔｒｉｇｇｅｒオブジェクトのリストはＡＮＤが取られ、単独のトリガとして定義される。ｃｌｎｔ＿ｔｒｉｇｇｅｒクラスは他のサブシステムからの次の要求を処理する。

【0207】

提供されるサービス	説明
GetMeasure（測定項目を得る）	選択された測定項目を返す。
SetMeasure（測定項目を設定する）	選択された測定項目を更新する。
GetOperator（演算子を得る）	選択された演算子を返す。

SetOperator（演算子を設定する）	選択された演算子を更新する。
GetOperand1（オペランド１を得る）	第１オペランドの測定項目を返す。
GetOperand2（オペランド２を得る）	演算子が間にあるかないかによって第２オペランドの測定項目を返す。
SetOperand1 （オペランド１を設定する）	第１オペランドの測定項目を更新する
SetOperand2 （オペランド２を設定する）	演算子が間にあるかないかによって第２オペランドの測定項目を更新する。
GetValue1（値１を得る）	第１オペランドの値を返す。
GetValue2（値２を得る）	第２オペランドの値を返す。
SetValue1（値１を設定する）	第１オペランドの値を更新する。
SetValue2（値２を設定する）	演算子が間にあるかないかによって、第２オペランドの値を更新する。
Operator=（演算子＝）	そのオブジェクトを別のオブジェクトへコピーする。

【０２０８】ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔの各インスタンス１５２３は１つのｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＤｅｆｉｎｉｔｉｏｎオブジェクトを持っていなければならない。ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＳｃｈｅｄｕｌｅおよびｃｌｎｔ＿Ｔｒｉｇｇｅｒのオブジェクトはオプションである。クラスｃｌｎｔ＿ＩｎｆｏＦｒａｍｅ™のインスタンス１５２４はｃｌｎｔＭａｎａｇｅｒオブジェクトによってインスタンシエートされる（保存されたアナリストをディスクから復元する時、または新しいフレームをサーバのＡＰＩから受け取った時）。他のサブシステムは普通はそれぞれのフォルダからそれらを検索することによって（ｃｌｎｔ＿Ｆｏｌｄｅｒ：：ＧｅｔＩｎｆｏＦｒａｍｅ™（））、ＩｎｆｏＦｒａｍｅ™オブジェクトにアクセスする。ｃｌｎｔＩｎｆｏＦｒａｍｅ™のインスタンスはｃｌｎｔ＿Ｍａｎａｇｅｒオブジェクトによってのみ削除される。ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅ™クラスは他のサブシステムからの次の要求を処理する。

【０２０９】

提供されるサービス	説明
GetName（名前を得る）	このＩｎｆｏＦｒａｍｅ™の名前をす。
GetHTMLFile （ＨＴＭＬファイルを得る）	ＨＴＭＬのフレーム・データを含んいるファイルの名前を返す。
UpdateHTMLFile （ＨＴＭＬファイルを更新する）	ＩｎｆｏＦｒａｍｅ™に対してそのータ・ファイルが更新されたことを知する（アノーテーション、ドリルダウン、グラフ作成に必要となる場がある）。

【０２１０】サーバのＡＰＩサブシステム１４０３はＭＤＴサーバ３２（ＤＡＩ１４）との通信を必要とするすべての関数をカプセル化する。これはユーザ・インターフェース１４０１およびフォルダ・マネージャ１４０２をクライアント－サーバ・インターフェースについての知識から隔離し、それらが小変更によっては影響されないようにする。図２２から分かるように、サーバのＡＰＩ１４０３は4つのＡＰＩモジュール、すなわち、Ｍｅｔａｄａｔａ™１５３１、ＩｎｆｏＦｒａｍｅ™作成１５３２、データ・ウェアハウス１５３３、およびセッション・マネージメント１４０３に分けることができる。また、サーバのＡＰＩサブシステム１４０３はＣＳＭ１４０４に対して通信するための一組の内部ルーチンも含んでいる。それらは4つのＡＰＩによって共有されている。各モジュールについて以下に説明する。

【０２１１】Ｍｅｔａｄａｔａ™のＡＰＩ１５３１はＭＤＴのＭｅｔａｄａｔａ™２５の一部分を表示したり変更したりするためのクライアント１２のすべての要求を処理する。この場合もＭｅｔａｄａｔａ™２５はサーバ３４上に常駐し、クライアント１２は必要に応じてＤＡＩ１４経由で、Ｍｅｔａｄａｔａ™２５の部分を検索する。Ｍｅｔａｄａｔａ™のＡＰＩ１５３１は他のクライアント・サブシステムに対して次のサービスを提供する。

【０２１２】

提供されるサービス	説明
GetDimensions （ディメンションを得る）	すべてのディメンションを表しているclnt_Dimensionオブジェクトのリストを返す。
AddDimensions （ディメンションを追加する）	新しいディメンションを追加する。

UpdateDimension （ディメンションを更新する）	既存のディメンションを更新する。
DeleteDimension （ディメンションを削除する）	既存のディメンションを削除する。
GetDimensionPartitions （ディメンションのパーティションを得る）	与えられたディメンションの中のすべての子パーティションに対するclnt_Partitionオブジェクトのリストを返す。
GetPartitions ディメンションのパーティションを得る）	与えられたディメンションの中のすべての子パーティションに対するclnt_Partitionオブジェクトのリストを返す。
AddPartition （パーティションを追加する）	新しいパーティションを追加する。
UpdatePartition （パーティションを更新する）	既存のパーティションを更新する。
DeletePartition （パーティションを削除する）	既存のパーティションを取り除く。
GetSegments（セグメントを得る）	与えられたパーティションの中の定義されたすべてのセグメントに対するclnt_Segmentオブジェクトのリストを返す。

AddSegment （セグメントを追加する）	新しいセグメントを追加する。
UpdateSegment （セグメントを更新する）	既存のセグメントを更新する。
DeleteSegment （セグメントを削除する）	既存のセグメントを取り除く。
GetMeasures（測定項目を得る）	３つのリスト、すなわち、基本測定項目、複合測定項目、およびユーザがアクセスできるすべての測定項目のリストを返す（いくつかの測定項目はクライアントのコードの中で内部的に使われている）。
AddMeasure （測定項目を追加する）	新しい測定項目を追加する。
UpdateMeasure （測定項目を更新する）	既存の測定項目を変更する。
DeleteMeasure （測定項目を削除する）	既存の測定項目を取り除く。
GetMeasureRelationship （測定項目間の関係を得る）	測定項目間の関係を検索する。

AddMeasureRelationship （測定項目間の関係を追加する）	新しい測定項目間の関係を追加する。
UpdateMeasureRelationship （測定項目間の関係を更新する）	既存の測定項目間の関係を更新する。
DeleteMeasureRelationship （測定項目間の関係を削除する）	既存の測定項目間の関係を取り除く。
GetRelationships（関係を得る）	与えられた属性に対する可能な関係を検索する。
GetRange（関係を得る）	与えられた属性に対する値の範囲を検索する。

【0213】MetadataTMのAPIはCSM1404と通信するために通信サービス・モジュール（下記参照）を使用する。いくつかのクラスが共同して上記の表にリストされているサービスのセットを提供する。クラスclnt_Dimensionはパブリック・メソッド、すなわち、GetDimensions（スタティックなクラス・メソッド）、AddDimension、UpdateDimension、DeleteDimension、およびGetDimensionPartitions、AddDimensionPartition、DeleteDimensionPartitionを有している。クラスclnt_Partitionはパブリック・メソッド、すなわち、GetSegments、AddSegment、DeleteSegment、UpdatePartitionを有している。クラスclnt_Segmentはパブリッタ・セグメント、すなわち、GetPartitions、AddPartition、DeletePartition、UpdateSegmentを有している。測定項目関数は2つのクラス、すなわち、clnt_BasicMeasureおよびclnt_CompositeMeasureによって表される。

【0214】InfoFrameTM作成API1532はDAI14が1つのアナリストを実行またはスケジュールすることを要求すること、およびDAI14からステータスおよび完了したInfoFrameTMを呼び出すことに関連したすべての関数を含んでいる。このAPI1532の中の関数はマネージャ・サブシステム1402およびユーザ・インターフェースサブシステム1401によって使われる。InfoFrameTM作成API1532は次のサービスを他のクライアント・サブシステムに対して提供する。

【0215】

提供されるサービス	説明
GetInfoFrameGenerationInstance（ＩｎｆｏＦｒａｍｅ作成のインスタンスを得る）	clnt_InfoFrameGenerationの１つ、そしてただ１つのインスタンスに対するポインタを返す。
GetStatus（ステータスを得る）	現在ペンディングになっている、および／または完了したInfoFrame™リストについてDAIへ問い合わせる
RetrieveFrame（フレームを検索する）	完成された特定のInfoFrame™をDAIから検索する。
RunAnalystNow（アナリストをすぐに実行する）	ただちに処理するためにDAIに対しInfoFrame™作成要求を送信する。
SchedleAnalyst（アナリストをスケジュールする）	InfoFrame™作成要求をDAIに対してそれを実行するスケジュール付きで送信する。
UpdateAnalyst（アナリストを更新する）	既存のスケジュールされたアナリストに対する変更をDAIに送信する。
CancelAnalyst（アナリストをキャンセルする）	以前にスケジュールされたInfoFrame™作成要求をキャンセルする。

【０２１６】ＩｎｆｏＦｒａｍｅ™のＡＰＩ１５３２もＣＳＭ１４０４と通信するために通信サービス・モジュールを使用する。上記のＩｎｆｏＦｒａｍｅ™作成要求ＡＰＩ１５３２関数はｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＧｅｎｅｒａｔｉｏｎクラスのパブリック・メンバーである。ｃｌｎｔ＿ＩｎｆｏＦｒａｍｅＧｅｎｅｒａｔｉｏｎクラスは「シングルトン」パターン（前に説明された）を使って、一度だけインスタンシエートされる。データ・ウェアハウスのＡＰＩ１５３３はデータ・ウェアハウス２４のスキーマに関する情報にＭＤＴＡ（アドミニストレータ）がアクセスする必要がある時点で、ＭｅｔａｄａｔａＴＭ２５のセットアップに関連したサービスを提供する。このＡＰＩ１５３３はｃｌｎｔ＿Ｓｃｈｅｍａクラスの中にカプセル化されている。データ・ウェアハウスのＡＰＩ１５３３は他のＭＤＴクライアント・サブシステムに対して次のサービスを提供する。

【０２１７】

提供されるサービス	説明
GetTables（テーブルを得る）	現在のスキーマ・データからのテーブルの名前にリストを返す。
GetColumns（カラムを得る）	与えられたテーブルからのカラムの名前のリストを返す。
GetPrimaryKeys（一次キーを得る）	与えられたテーブルにおける一次キーの名前のリストを返す。
GetForeignKeys（外部キーを得る）	与えられたテーブルにおける外部キーのリストを返す。
ForeignToPrimary（外部キーから一次キーへ）	与えられたテーブルの中の与えられた外部キーに関連付けられた一次キーのリストを返す。
PrimaryToForeign（一次キーから外部キーへ）	与えられたテーブルの中の与えられた一次キーに関連付けられた外部キーのリストを返す。
LoadSchema（スキーマをロードする）	サーバからスキーマをロードする。成功した場合、Ｔｒｕｅを返す。

【０２１８】データ・ウェアハウスのＡＰＩ１５３３もＣＳＭ１４０４と通信するために通信サービス・モジュールを使用する。データ・ウェアハウスのＡＰＩ１５３３はｃｌｎｔ＿Ｓｃｈｅｍａクラスの中にカプセル化されている。ｃｌｎｔ＿Ｓｃｈｅｍａクラスは上記の表の中に記載されたＡＰＩ呼出しに直接対応するパブリック・メンバー関数を備えている。ＬｏａｄＳｃｈｅｍａ関数は他のＡＰＩ関数がアクセスするためにデータ・ウェアハウスのスキーマのすべてをクライアント上にロードする。そのスキーマは使用のたびに捨てられる。

【０２１９】セッション・マネジメントＡＰＩ１５３４はＭＤＴサーバとのセッションを確立すること（および脱出時にそのコネクションを閉じること）に関連する関数を含んでいる。これはユーザおよびパスワードのマネージメントに関連する関数を含む。セッション・マネジメントＡＰＩは他のＭＤＴクライアント・サブシステムに対して次のサービスを提供する。

【０２２０】

提供されるサービス	説明
GetSessionManager （セッション・マネージャを得る）	クラスclnt_SessionManagerの１つそしてただ１つのインスタンスに対するポインタを返す。
Connect（接続）	サーバに対するコネクションを確立し、与えられたユーザ名およびパスワードを認証しようと試みる。
Disconnect（切離し）	サーバに対するコネクションを順序だててシャットダウンする。
UpdateUserPassword （ユーザのパスワードを更新する）	サーバ上のユーザのパスワードを変更する。

【０２２１】セッション・マネジメントＡＰＩ１５３４もＣＳＭ１４０４と通信するために通信サービス・モジュールを使用する。上記のセッション・マネジメント関数はｃｌｎｔ＿ＳｅｓｓｉｏｎＭａｎａｇｅｒのパブリック・メンバー関数である。クラスｃｌｎｔ＿ＳｅｓｓｉｏｎＭａｎａｇｅｒは「シングルトン」パターンを使って、一度、そしてただ一度だけインスタンシエートされる。

【０２２２】通信サービスはＣＳＭ１４０４経由でＤＡＩ１４に対して通信するための関数をカプセル化している。これらの関数はサーバのＡＰＩサブシステム１４０３の中の4つのＡＰＩ　１５３１、１５３２、１５３３および１５３４によって共有されている。その通信サービスの機能はサーバのＡＰＩサブシステム１４０３の中の他のすべてのクラスに対するプライベートのスーパークラスとなる、クラスｃｌｎｔ＿Ｃｏｍｍｕｎｉｃａｔｉｏｎｓの中にカプセル化されている。

【０２２３】＜３．データ抽象化インテリジェンス（ＤＡＩ）サブシステム１４＞データ抽象化インテリジェンス・サブシステム１４が、以下にさらに詳しく説明される。本発明（Ｍａｎａｇｅｍｅｎｔ　Ｄｉｓｃｏｖｅｒｙ　Ｔｏｏｌ™またはＭＤＴとも呼ばれる）の重要な特徴は、それによってユーザ１４０５が問題のオブジェクトのビューイングおよび理解における詳細のレベルを容易に選定できること、そしてＭＤＴによって作成されるＩｎｆｏＦｒａｍｅ™の中に反映されるこれらの異なるレベルを有することである。例えば、ユーザ１４０５が「製品」の概念について考える時、その人はその企業全体によって販売されているすべての製品を意味する場合もあり、あるいは、全製品のうちの関心のあるいくつかのサブセットを意味する場合もある。このサブセットは製品の属性についての制限、すなわち、例えば、「男性用の衣類」などの単独の制限（「製品部門＝男性用衣類」のように定義される）、あるいは「デザイナーＹによって作られた高価な男性用スーツ」などの複数の制限（部門、価格、および製造者についての制限によって同様に定義される）のいずれかによって定義することができる。

【０２２４】本発明の設計における洞察の１つはそのようなサブセットは階層を形成すること、そしてこれらの階層は、ユーザがＩｎｆｏＦｒａｍｅ™の製品に対する関連の粒度レベルについて考え、選択することの両方のために、便利で強力な方法を提供することができるという考えである。１つの重要な技術的ポイントであるが、ユーザ１４０５からは部分的に隠されている点は、関連付けられているセグメントがどのようにパーティションを形成するかということである。

【０２２５】本発明のＭａｎａｇｅｍｅｎｔ　Ｄｉｓｃｏｖｅｒｙ　Ｔｏｏｌ™（管理発見ツール）は単独の論理的な企業データの首尾一貫したビューであるデータ・ウェアハウス２４のトップにある。普通、データを格納するには多くの異なった方法がある。すなわち、与えられたデータの集合に対して異なるテーブル構造またはスキーマがある。すべてではないにしても、ほとんどの企業において、一組の基本的なデータ・タイプの小集合があり、それは最低レベルの粒度であって、代表的なものは製品、顧客、トランザクションなどの特定のエンティティである。これらのエンティティには一組の属性が

付随しており、そしてそれらの属性には値があると考えることができる。リレーショナル・モデルにおいては、これは属性がカラムにマップされている、エンティティのテーブルに対応する。この場合も、物理的には、それらをいくつかのテーブルとして格納することができる。しかし、概念的には、これがＭＤＴが作業対象である。

【０２２６】図２３は部門属性１６０１を選んでいるユーザ１４０５によって形成された階層を示している。ユーザは「部門」属性１６０１を選んでいて、次に属性のセグメント「男性用の衣類部門」１６０２を選択し、「製品のタイプ」属性１６０３を選び、次にその属性のセグメント「シャツ」１６０４を選択し、「製造者」属性１６０５、次に「デザイナーＸ」１６０６、そして最後に、「サイズ」属性１６０７、および特定のパーティション１６０８／１６０９を選んでいる。対象とする最終「セグメント」は「デザイナーＸの男性用シャツ」である。このセグメントに到達するにはいくつかの方法があり、特に関連の属性の順序は任意でよいことに留意されたい。

【０２２７】このスキームによってＭＤＴの他の部分に対するいくつかの要求条件が生成される。まず第１に、各ディメンションの属性が利用できなければならない。第２に、各属性の合法の値も利用できなければならない。有限のドメインに対して、これらはユーザ１４０５に対してリストされなければならない。有限のドメインに対して、その時点での最小値および最大値が役に立つ可能性がある。ここで最初におそらく回避できる微妙な問題がある。可能な属性値は、実際には、すべてのケースにおいて適切であるとは限らない。上記の例の中で、すべての製造者が「男性用のシャツ」を作っているとは限らない。合法の値についてデータベースに問い合わせるか、あるいはこの情報を追加のＭｅｔａｄａｔａ™として格納することは可能である。

【０２２８】本発明の重要な条件はユーザ定義のセグメントを保存および再使用する機能を提供することである。「デザイナーＸの男性用シャツ」がユーザ１４０５にとって重要なカテゴリーである場合、そのユーザはそのカテゴリーを保存し、それをＩｎｆｏＦｒａｍｅ™の作成において再使用できる必要がある。この方式によってユーザはセグメントを定義し、そしてこれらのセグメントを階層の中に自動的にキープしておくことができる。下記の表は一組の名前付きセグメントおよびそれらの定義、すなわち、それらの属性についての制限を提供している。すべてのセグメントはディメンション「製品」上にある。

【０２２９】

名前	部門	メーカ	タイプ	サイズ
男性用衣類	ＭＣ			
男性用シャツ	ＭＣ		シャツ	
デザイナーＸのシャツ		デザイナーＸ	シャツ	
デザイナーＸの男性用シャツ	ＭＣ	デザイナーＸ	シャツ	

男性用パンツ	ＭＣ		パンツ	
大型の男性用シャツ	ＭＣ		シャツ	大型
大型の男性用パンツ	ＭＣ		パンツ	大型
Ｇａｐの製品		Ｇａｐ		
デザイナーＸの製品		デザイナーＸ		
Ｇｕｅｓｓの製品		Ｇｕｅｓｓ		
中型および大型の男性用衣類	ＭＣ	中型＋大型		
小型の男性用衣類	ＭＣ			小型
小型の男性用シャツ	ＭＣ		シャツ	小型

【０２３０】上記のセグメントによって図２４に示されているセグメント階層が発生する。この構造には本来的な曖昧さがあることに留意されたい。与えられたセグメントはいくつかの異なる階層に属する可能性があり、そしてその下に子があってそれらがいくつかの異なるパーティションの中にある。例えば、セグメント「デザイナーＸ－男性用シャツ」は２つのパーティションに属している。最初のパーティションは「デザイナーＸ－シャツ」の上にあり、さらに区別するために「男性用」という値を使う（「その他」のセグメントはここでは「デザイナーＸのシャツであるが、男性用のシャツではない」という意味になることに留意されたい）。このセグメントは「男性用のシャツ」の中のパーティションの一部でもあり、製造者の属性がデザイナーＸであると制限する。ここで、「その他」のセグメントは「デザイナーＸ以外の製造者によって作られた男性用シャツ」を意味する。

【０２３１】曖昧さを解消するために、パーティションの概念が明確にされなければならない。したがって、図２４を図２５に示されているように書き直してパーティション（暗い境界の）および「その他」のセグメントを含める。これで図２５から「パーティション」の概念が「セグメント」の概念と同様に重要であることが分かる。実際、この２つの概念は分離できない。

【０２３２】図１に戻ると、ＭＤＴデータ抽象化インテリジェンス（ＤＡＩ）サブシステム１４はクライアント・サブシステム１２と「データおよびスキーマ操作」（ＤＳＭ）サブシステム１６との間に入っている。ＤＡＩ１４（およびＤＳＭ１６）はアプリケーション・サーバ３２上でデスクトップ３０とデータ・ウェアハウス２４との間にある。ＤＡＩ１４はユーザが選択したＩｎｆｏＦｒａｍｅ™をインスタンシエートすること、そしてこのインスタンシエーションにおいて使われる数種類のＭｅｔａｄａｔａ™を管理することを担当する。このＭｅｔａｄａｔａ™は、（１）データ・ウェアハウスの中のリレーショナル・データのカスタマイズ可能な「ディメンション化」を提供するディメンションおよび測定項目、（２）ＩｎｆｏＦｒａｍｅ™の中に説明的なテキストを提供する測定項目間の関係、および（３）時間に関連付けられたＭｅｔａｄａｔａ™を表す。また、ＤＡＩ１４はクライアント層１２から発生したこのＭｅｔａｄａｔａ™に対する更新を処理し、また、いくつかの他の種類の更新を、主としてそれらをＤＳＭ１６サブシステムへ通して渡すことによって処理する。最後に、ＤＡＩ１４はクライアント１２またはスケジューラ・サブシステム１８からのアラート要求を実行する。

【０２３３】総合的なクライアント／サーバの設計は、２種類のＤＡＩサブシステム１４を想定する。第１の種類のＤＡＩ１４はクライアント１２からの情報に対する連続的な、同期要求を処理するために必要である。例えば、クライアント１２はアクティブなＭＤＴセッションの間に情報に対する多くの要求を行う可能性がある。Ｓ－ＤＡＩ（シリアルＤＡＩの場合）がこれらの要求を処理する。第２の種類のＤＡＩ１４はトリガの定義または

ＩｎｆｏＦｒａｍｅ™の定義を含む可能性のあるアナリストを実行する。この実行は比較的長い時間（例えば、数時間）掛かる可能性があり、そして同時並行的（コンカレント）に実行される。このタイプのＤＡＩ１４はコンカレントＤＡＩという意味でＣ－ＤＡＩと呼ばれることがある。Ｃ－ＤＡＩはＩｎｆｏＦｒａｍｅ™要求に応答してＳ－ＤＡＩによってスポーン（ｓｐａｗｎ）され、それに必要なすべての情報が与えられる。終了すると、それはその結果のＩｎｆｏＦｒａｍｅ™をサーバのディスク・ファイルに書き込む。

【０２３４】図２６はＤＡＩ１４および他のサブシステムと本発明のＭＤＴのコンポーネントとの間の、一般的な、高レベルのデータ・フローを示している。ユーザが本発明のシステムにログオンすると、シリアルＤＡＩ（Ｓ－ＤＡＩ）１４Ａが各種の要求をサービスするために生成される。クライアント１２からのすべてのユーザ要求はクライアント／サーバ・モジュール（ＣＳＭ）１４０４を通して流れ、このＣＳＭはクライアント１２とサーバ３２との間の通信を制御する。Ｍｅｔａｄａｔａ™に対する要求は、読出しおよび更新（書込み）要求の両方とも、Ｓ－ＤＡＩによって処理される。読出し要求によって、Ｓ－ＤＡＩはＭＤＴのＭｅｔａｄａｔａ™のテーブルの中に含まれているＭｅｔａｄａｔａ™に対するデータおよびスキーマ操作（ＤＳＭ）モジュール１６に問い合わせる。Ｓ－ＤＡＩはＬｕｃｅｎｔ Ｔｅｃｈｎｏｌｏｇｉｅｓ社のＢｅｌｌ Ｌａｂｏｒａｔｏｒｉｅｓから入手できる、階層データ構造を管理するためのシステムである「Ｃｌａｓｓｉｃサブシステム」を使うことによって更新要求を処理する。Ｃｌａｓｓｉｃサブシステムはセグメント階層の中の新しいユーザ・セグメントをチェックして適切に配置するために使うことができる。また、シリアルＤＡＩは以前に作成されたＩｎｆｏＦｒａｍｅ™に対する要求を、サーバのディスクからフェッチすることによって処理する。

【０２３５】アナリストを生成することをユーザが要求すると、コンカレントＤＡＩ１４Ｂがディスパッチャによって生成され（リソースが許す場合）、そしてそのコンカレントＤＡＩ１４Ｂにアナリストの定義の中ですべての情報が渡される。コンカレントＤＡＩ１４Ｂは次にその組込みアルゴリズムおよびＣＳＭ１４０４から要求されたＭｅｔａｄａｔａ™を使ってＩｎｆｏＦｒａｍｅ™を作成し、次にそれはサーバのディスク上に格納される。ＣＳＭモジュール１４０４はメッセージを１つのプロセスから別のプロセスへ渡す低レベルのサービスを提供する。或るクラスは使い易くするために、以下でさらに詳しく説明されるように、ＣＳＭ１４０４のトップに構築される。

【０２３６】このメッセージ・パッシングの設計は５つのクラスおよび１つの列挙型のｍｄｔ＿ＭｅｓｓａｇｅＴｙｐｅから構成される。列挙型はＭＤＴの中の各タイプのためのユニークな値を持つ。ｍｄｔ＿ＴＩｎＳｔｒｅａｍＨａｎｄｌｅおよびｍｄｔ＿ＴＯｕｔＳｔｒｅａｍＨａｎｄｌｅはそれぞれ入力および出力のメッセージ・ストリームのハンドルである。ｍｄｔ＿ＭｅｓｓａｇｅはすべてのＭＤＴメッセージに対する抽象ベース・クラスである。ｄａｉ＿ＭａｓｓａｇｅＨａｎｄｌｅｒはメッセージを処理することができるオブジェクトに対する抽象ベース・クラスである。最後に、ｄａｉ＿ＭｅｓｓａｇｅＲｅｇｉｓｔｒｙはメッセージ・ハンドラのレジストリである。

【０２３７】ｍｄｔ＿ＭｅｓｓａｇｅＴｙｐｅは以下に定義される。

```
enum mdt_MessageType {
    UNDEFINED,
    CS_LOGIN,
    SC_LOGIN_RESPONSE,
    CS_GENERATE_INFOFRAME,
    CS_GET_INFOFRAME_STATUS,
    SC_INFOFRAME_STATUS,
    CS_GET_INFOFRAME,
    SC_INFOFRAME,
    END_OF_ENUM
};
```

【０２３８】ｍｄｔ＿ＭｅｓｓａｇｅＴｙｐｅはＭＤＴによって理解されるすべてのメッセージ・タイプを定義するｅｎｕｍである。ｍｄｔ＿Ｍｅｓｓａｇｅから導かれる各クラスに対して、ｍｄｔ＿ＭｅｓｓａｇｅＴｙｐｅの中に少なくとも１つの値がなければならない。このｍｄｔ＿ＭｅｓｓａｇｅＴｙｐｅの宣言はＭＤＴのすべてのコンポーネントによって使われる。示されているメッセージ・タイプは一例に過ぎない。以下に現在定義されているメッセージ・タイプは単なる１つの例である。この定義はＭＤＴコードが書かれているように見える。

【０２３９】ｍｄｔ＿ＴＩｎＳｔｒｅａｍＨａｎｄｌｅは以下のように定義される。

```
class mdt_TInStreamHandle:public cs
m_InStreamHandle{
```

```
  public:
   mdt_TInStreamHandle():d_mtype(UND
EFINED){}
   virtual void Connect(RWvistream*)
;
   mdt_MessageType GetMessageType()
const;
   private:
    mdt_MessageType d_mtype;
 };
```

【0240】mdt_TInStreamHandleは入って来る（受信される）メッセージ・ストリームに対する、タイプが決められたハンドルである。mdt_TInStreamHandleはCSMモジュール1404から入って来るメッセージ・ストリームをピックアップするために使われる。mdt_TInStreamHandleは入って来るメッセージ・ストリームからそのメッセージのタイプを自動的に読み、そのメッセージ・タイプを得るためのメンバー関数を提供する。mdt_MessageRegistryのユーザはこのクラスを使う必要はない。

【0241】mdt_TOutStreamHandleは以下に定義される。

```
 class mdt_TOutStreamHandle:public c
sm_OutStreamHandle{
  public:
   mdt_TOutStreamHandle(mdt_MessageT
ype t):d_mtype(t){}
   virtual void Connect(RWvostream*)
;
   mdt_MessageType GetMessageType()
const;
  private:
   mdt_TOutStreamHandle();
   mdt_MessageType d_mtype;
 };
```

【0242】mdt_TOutStreamHandleは出て行くストリームに対する、タイプの決められたハンドルである。mdt_TOutStreamHandleはCSM1404を経由してストリーム型のメッセージを送信するために使われる。そのmdt_TOutStreamHandleはメッセージ・タイプを有し、受信端においてmdt_TInStreamHandleがそのメッセージ・タイプを解読できるようにそのストリームに対して自分のメッセージ・タイプを自動的に書き込む。

【0243】mdt_Messageは以下に定義される。

```
 class mdt_Message{
  public:
   mdt_Message():d_restoreVersion(0)
, d_restoreFlag(false){}
   mdt_Message(const mdt_Message &);
    virtual mdt_MessageType GetMessa
geType() const;
   virtual int GetLastVersion() cons
t=0;
   virtual void SetRestoreVersion(in
t restoreVersion);
   virtual int GetRestoreVersion() c
onst;
   virtual void saveGuts(RWvostream
```

```
&)  const;
    virtual void RestoreGuts(RWvistre
am &);
    virtual bool IsRestored() const;
  protected:
    virtual void SetRestoredFlag(bool
);
  private:
    bool d_restoreFlag;
    int d_restoreVersion;
};
```

【０２４４】要求および応答はクライアント１２、シリアルＤＡＩ１４Ａ、コンカレントＤＡＩ１４Ｂ、スケジューラ１８と、ディスパッチ２５１３との間で、クライアント／サーバ・モジュール（ＣＳＭ）１４０４によって通信される。ＣＳＭ１４０４は要求または応答を送信または受信するために使われるｍｄｔＭｅｓｓａｇｅクラスを実装する。受信のｍｄｔ＿Ｍｅｓｓａｇｅは入力ストリームを含んでいる。送信のｍｄｔ＿Ｍｅｓｓａｇｅは出力ストリームを含んでいる。ストリームはよく知られているＣ＋＋の構造体であり、これによってユーザは長いメッセージの要素をバッファの中に「ストリーム」するか、あるいは、メッセージの要素をバッファからストリームとして出すことができる。

【０２４５】このＭＤＴの発明のために実装される各要求および応答は、ｍｄｔ＿Ｍｅｓｓａｇｅの基本クラスから導かれるメッセージによって表され、そして自分自身をストリーム・インまたはストリーム・アウトするための適切な関数を有している。ＣＳＭ１４０４はｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔおよびｃｓｍ＿ＳｅｒｖｅｒＳｏｃｋｅｔクラスを実装する。各タイプのソケットはＴＣＰ／ＩＰソケットを含んでいる。ＴＣＰ／ＩＰソケットはＴＣＰ／ＩＰネットワークに対するよく知られたＡＰＩである。他の実装が考えられる。各タイプのソケットはメッセージからそのストリーム・バッファを抽出することができ、そしてそれをソケット経由で受信のｃｓｍ＿Ｓｉｍｐｌｅまたはｃｓｍ＿ＳｅｒｖｅｒＳｏｃｋｅｔへ送るか、あるいは、送信のｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔからストリーム・バッファを受取り、それをメッセージの中にインストールすることができる。

【０２４６】ｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔはＭＤＴサブシステム間で約束された１対１の関係でメッセージを通信するために使われる。ｃｓｍ＿ＳｅｒｖｅｒＳｏｃｋｅｔはＭＤＴサブシステムが他の多くのサブシステムからメッセージを受け付けることができるようにするために使われる。図３２において、クライアント・サブシステム１２はｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔを維持し、それを使ってマスタ・サブシステム２５１１からＳＤＡＩサブシステム２５１２を要求し、またその後、それを使ってそのＳＤＡＩサブシステムとメッセージを交換する。クライアント１２は、クライアント１２に対するユーザの入力がサーバとの交換を必要とする時はいつでもこのソケットを使用する。

【０２４７】図３２において、マスタ・サブシステム２５１１はＳＤＡＩサブシステム１４Ａを要求したい任意のクライアント・サブシステム１２からのメッセージを受信するために、ｃｓｍ＿ＳｅｒｖｅｒＳｏｃｋｅｔを維持する。マスタがアクティブにＳＤＡＩを開始するか、さもなければＳＤＡＩに対して面倒をみている時以外は、それはｃｓｍ＿ＳｅｒｖｅｒＳｏｃｋｅｔを聴取している。ＳＤＡＩサブシステム２５１１はクライアント・サブシステム１２とメッセージを交換するために、ｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔを維持する。それがクライアントの要求を実際に実装している時を除いて、ＳＤＡＩ１４Ａはそのｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔを聴取している。

【０２４８】ユーザ１４０５がただちに実行するためにアナリストをサブミットしたとき、ＳＤＡＩサブシステム１４Ａはそのアナリストをディスパッチャ２５０１へ通信するために新しいｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔを構築する。ユーザがスケジュール型の、あるいはトリガ型のアナリストをサブミットすると、ＳＤＡＩはそのアナリストをスケジューラ１８に対して通信するために１つのｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔを構築する。スケジューラ１８は任意の、またはすべてのＳＤＡＩサブシステム２５１１からスケジュール型の、あるいはトリガ型のアナリストを収集するために、ｃｓｍ＿ＳｅｒｖｅｒＳｏｃｋｅｔを維持する。スケジューラがスケジュール型の、あるいはトリガ型のアナリストを立ち上げるべき時が来たと判断すると、スケジューラはｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔを１つ構築してそのアナリストをディスパッチャ・サブシステム２５１３に対して通信する。スケジューラがそのスケジュール型の、あるいはトリガ型のアナリストのリストをテストしている時、またはそれらをディスパッチしている時を除いて、スケジューラはそのｃｓｍ＿ＳｉｍｐｌｅＳｏｃｋｅｔを聴取している。

【０２４９】ディスパッチャ・サブシステム２５１３は

準備完了状態の任意のＳＤＡＩサブシステム２５１１から、またはスケジューラ・サブシステム１８から実行のためのアナリストを収集するため、あるいは、ＣＤＡＩサブシステム１４Ｂからのアナリスト要求を収集するために、１つのｃｓｍ_ＳｅｒｖｅｒＳｏｃｋｅｔを維持する。１つのＳＤＡＩまたはそのスケジューラが１つのアナリストを提示すると、ディスパッチャはその実行のためのリソースが利用できるようになるまで、それを保持する。リソースが利用できるようになると、ＳＤＡＩはＣＤＡＩ１４Ｂを開始させる。ＣＤＡＩがそのアナリストに対する要求を返すと、ＳＤＡＩは１つのｃｓｍ_ＳｉｍｐｌｅＳｏｃｋｅｔを生成して、そのアナリストをＣＤＡＩに対して通信する。ＣＤＡＩの管理を開始している時を除いて、ＳＤＡＩサブシステム２５１３はそのｃｓｍ_ＳｅｒｖｅｒＳｏｃｋｅｔを聴取している。ＣＤＡＩサブシステム１４Ｂはディスパッチャ・サブシステム２５１３からそのアナリストを収集するために始動された後すぐに、ｃｓｍ_ＳｉｍｐｌｅＳｏｃｋｅｔを１つ構築する。このメッセージを収集した後、そのｃｓｍ_ＳｉｍｐｌｅＳｏｃｋｅｔを捨てる。ＣＤＡＩサブシステム１４Ｂは本発明の他のサブシステムとはメッセージを交換しない。

【０２５０】ｍｄｔ_Ｍｅｓｓａｇｅ抽象基本クラスはＭＤＴのプロセス間メッセージの内容を保持するオブジェクトを定義する。各メッセージはメッセージ・タイプ、メッセージ・バージョン、およびメッセージ・データのストリームとの間でそのデータを読み書きする機能を有する。各タイプのメッセージに対して、ｍｄｔ_Ｍｅｓｓａｇｅから導かれたクラスの具体的な実装がある。各メッセージ・クラスはＧｅｔＬａｔｅｓｔＶｅｒｓｉｏｎを実装して、そのバージョンを返さなければならず、そしてＧｅｔＭｅｓｓａｇｅＴｙｐｅを実装してそのｍｄｔ_ＭｅｓｓａｇｅＴｙｐｅを返さなければならない。また、Ｒｏｇｕｅ Ｗａｖｅ ｓａｖｅＧｕｔｓおよびｒｅｓｔｏｒｅＧｕｔｓメソッドを実装してその永続的なメンバー・データをストリームへ書込み、そしてそのメンバー・データをストリームから読み戻さなければならない。アンパッキングの順序は「先入れ先出し」である。メッセージ・タイプあたりに導かれたメッセージ・クラスが１つだけあるが、導かれたメッセージ・クラスによって使われるメッセージ・タイプはいくつかあり得る。同じメッセージ・コードが送信と受信の両方のプロセスにリンクされる必要がある。送信のプロセスと受信のプロセスにおけるコードのバージョン間のミスマッチを誤りなく確実に処理するために、バージョン・チェックが使われる。そのメッセージ・バージョンは到来するメッセージ・ストリームを復元できることを保証するため、そして、そのクラスの古いバージョンによって保存されたストリームをマイグレートするために、ｒｅｓｔｏｒｅＧｕｔｓによって使われる。ｍｄｔ_Ｍｅｓｓａｇｅｓは上記で定義されたｍｄｔ_ＴＯｕｔＳｔｒｅａｍＨａｎｄｌｅ／ｍｄｔ_ＴＩｎＳｔｒｅａｍＨａｎｄｌｅによってポイントされるストリームとの間で、その中身を保存／復元することによって、ＣＳＭモジュール１４０４からｍｄｔ_Ｍｅｓｓａｇｅが送信／受信される。

【０２５１】ｄａｉ_ＭｅｓｓａｇｅＨａｎｄｌｅｒは以下に定義される。

```
class dai_MessageHandler{
  public:
    virtual bool HandleMessage(const
mdt_Message &)=0;
  };
```

【０２５２】ｄａｉ_ＭｅｓｓａｇｅＨａｎｄｌｅｒはメッセージ・ハンドラ・クラスに対する抽象基本クラスである。プロセスが受信したメッセージをディスパッチするためにメッセージ・レジストリを使う場合、少なくとも１つの具体的なメッセージ・ハンドラ・クラスが実装されなければならない。登録されたメッセージあたり１つ程度の数のメッセージ・ハンドラが実装される可能性がある。

【０２５３】ｄａｉ_ＭｅｓｓａｇｅＲｅｇｉｓｔｒｙは以下に定義される。

```
class dai_MessageRegistry{
  public:
    dai_MessageRegistry(csm_Connect &
);
    void DispatchMessage();
    void RegisterMessage(mdt_MessageT
ype, dai_MessageCreateFunc, dai_Messag
eHandler &);
  private:
    dai_MessageCreateFunc d_createFun
c;
```

```
        dai_MessageHandler *d_handler;
        csm_Connect *d_connect;
    };
```

【0254】dai_MessageRegistryはそれを使う各プロセスの中で一度だけインスタンシエートされることを意味するクラスである。そのメッセージ・レジストリは各メッセージ・タイプごとにハンドラを登録するためのメソッド、および到来するすべてのメソッドをディスパッチするためのメソッドを提供する。そのディスパッチ・メソッドはそのアプリケーションのメイン・ループのように働く。そのハンドラのHandleMessageメソッドの戻り値は、メッセージが処理された後でDispatchMessageがブロックするか戻るかどうかを決定する。戻り値が「真」（true）であった場合、それはブロックする。

【0255】次は、MDTのタイプのストリーム・ハンドルおよびメッセージ・レジストリを使って1つのプロセスから別のプロセスへメッセージが送信されるときに発生するイベントのセットである。

1．　送信側のプロセスは具体的なメッセージ・クラスのインスタンス（mdt_Messageから導かれる）、MSを構築し、そしてそれを適切なメッセージ・データと一緒にロードする。

2．　送信側のプロセスはそのメッセージと同じメッセージ・タイプのmdtTOutStreamHandleオブジェクトを生成する。このmdt_TOutStreamHandlcオブジェクトはそのメッセージ・タイプをそのストリームへ書き込む。

【0256】3．　送信側のプロセスはそのメッセージのsaveGutsメンバー関数を使ってそのメッセージをメッセージ・ストリームへ書き込む。

4．　メッセージのsaveGutsメソッドは最初にそのメッセージのバージョン番号をそのストリームに対して書き込む基本のクラス・メソッドを呼び出す。次に、そのメッセージの永続的なメンバーのデータ・フィールドをそのストリームへ保存する。

5．　送信側のプロセスはcsm_SendProcessConnect::Send()を呼び出して、そのメッセージ・ストリームを送る。

【0257】6．　CSMはそのストリーム・オブジェクトからバイトを抽出し、そのバイトを受信側のプロセスへ送る。

7．　mdt_TOutStreamHandleオブジェタトが破壊されると、それは自分が接続されていたストリーム・オブジェクトを破壊する。

8．　受信側のプロセスはcsm_ReceiveProcess::Receive()に対する呼出しにおいて待っている必要がある。内部的には、おそらくいくつかのキューが存在している。

9．　受信側のプロセスの中のCSMはキューの中の最も古いメッセージを得る。次にそれはそのバイトをストリーム・オブジェクトに変換し、そしてそのストリームをcsm_ReceiveProcess::Receive()に対して渡されたmdt_TInStreamHandleオブジェクトに接続する。

10．　ストリームがそのハンドルに接続されると、そのハンドルはそのメッセージ・タイプをそのストリームから読み、それを覚える。

【0258】11．　最後に、制御がcsm_ReceiveProcess::Receive()から呼出し側へ戻る。

12．　受信側のプロセスはそのメッセージ・タイプをmdt_TInStrcamHandleから取得し、そのメッセージ・タイプに対するメッセージ・クラスの正しいタイプの空のインスタンスを構築する。メッセージ・ディスパッチャが使用中の場合、これはdai_MessageRegistry::DispatchMessages()によって処理される。

13．　受信側のプロセスはそのメッセージのrestreGuts関数を呼び出す。メッセージ・ディスパッチャが使用中の場合、これはdai_MessageRegistry::DispatchMessages()によって処理される。

【0259】14．　そのメッセージのrestoreGutsメソッドはまず最初に、バージョン番号を読んでそれをRestoreVersionメンバーとして保存する自分の基本クラス（普通はmdt_Message）のrestoreGutsメソッドを呼び出す。次に、制御は導かれたバージョンのrestoreGutsへ戻る。それはGetRestoreVersionを呼び出し、その結果のバージョン番号を使ってどのデータ・メンバーをそのストリームから読み出すか、そしてどんな順序でそれらを読み込むかを決定する。次に、そのメッセージ・クラスのデータ・フィールドがそのストリームから読み戻される。

【0260】15．　受信されたメッセージはここではその最終のフォームになっている。ディスパッチャが使用中の場合、それはレジストリの中でこのメッセージ・タイプを探してそのハンドラをルックアップし、そしてそのHandleMessageメソッドを呼び出す。

16．　そのメッセージを受信するために使われたmdt_TInStreamHandleオブジェクトが破壊されている場合（あるいは、csm_ReceiveProcess::Receive()に対する別の呼出しによって再接続されていた場合）、それがポイントしていたストリームは削除される。

【0261】このMDTの発明によって、アナリストと

呼ばれる強力なレポート作成オブジェクトを定義することにより、他のユーザがそれぞれのデータ・ウェアハウス２４を監視することができる。アナリストは一般のタイプの分析のカプセル化されたものであり、一組のパラメータ、スケジュール、およびトリガ条件を提供することにより、ユーザ１４０５によってカスタマイズされたものである。そのアナリストはトリガ条件が成立しているかどうかについてそのトリガ条件を定期的にチェックしするか、あるいは、トリガ条件が成立している場合はそのスケジュール上で定期的に実行する。それ以外の場合、それはただちに実行する。ＩｎｆｏＦｒａｍｅ™は他のアナリストを実行するために使うことができ、関連付けられたＩｎｆｏＦｒａｍｅ™を作成することができる各種のデータおよびハイパーリンクを含んでいる。アナリストの定義およびＩｎｆｏＦｒａｍｅ™の作成を含めて、ＭＤＴの機能は、「ＭＤＴのＭｅｔａｄａｔａ™２５」と呼ぶデータ・ウェアハウスの「ＭＤＴビュー」を必要とする。

【０２６２】「Ｍｅｔａｄａｔａ™」という言葉は一般に、データベースまたはデータ・ウェアハウス２４（図１）の中の各種の「データに関するデータ」のことをいう。ＭＤＴのＭｅｔａｄａｔａ™２５はそのデータベースの関係構造によって制限されていないデータのカスタマイズ可能なビューを提供する。ＭＤＴのＭｅｔａｄａｔａ™２５は一組のテーブルとしてデータ・ウエアハウス２４の中に格納されており、スタートアップ時に本発明のＭＤＴによって読み出される。データ・ウェアハウス２４の中にＭｅｔａｄａｔａ™２５を埋めて置くことはＭＤＴのインストレーション・プロセスの１つの重要な部分であり、そしてＭＤＴアドミニストレータはデータ・ウェアハウス２４の構造の知識を使ってＭＤＴのＭｅｔａｄａｔａ™を拡張して維持することが一般に必要となる。

【０２６３】ＭＤＴのＭｅｔａｄａｔａ™には主な種類が４つある。それらは（１）ディメンションおよび属性、（２）セグメントおよびパーティション、（３）測定項目、（４）測定項目間の関係である。本発明の仕様は一連のテーブルによってそれぞれの種類のＭｅｔａｄａｔａ™を記述する。各テーブルはカラム名およびそのカラムの中のデータ値のタイプ（括弧内）を示す。いくつかのカラムはＭＤＴ指向のエンティティまたはオブジェクトを定義し、そして他のカラムはＭＤＴのＭｅｔａｄａｔａ™とウェアハウスの関係スキーマとの間の情報のマッピングを提供する。各テーブルの一次キーは斜体文字で書かれている（２つ以上のカラム名が斜体文字になっていた場合、それらのカラム名の組合わせがそのキーを形成する）。

【０２６４】ＭＤＴのＭｅｔａｄａｔａ™２５は、或るデータ属性は値の有限集合を持ち得るという明示的な概念を含む。これらは「列挙型（ｅｎｕｍｅｒａｔｅｄ）の属性」と呼ばれる。例えば、本発明は属性「州（Ｓｔａｔｅ）」は値の有限集合に限定される（すなわち、米国の５０の各州、そのデータ・ウェアハウスがどんな符号化を使うとしても）という事実を表すことができる。また、ＭＤＴのＭｅｔａｄａｔａ™のテーブルは、便宜的に、ソース・コードのヘッダ・ファイルの中で列挙型で表される値の集合を指す。「ｅｎｕｍ」という語はこのタイプのカラムの値を指す。

【０２６５】ＭＤＴのＭｅｔａｄａｔａ™２５のテーブル表現はまず最初に、主な種類の各Ｍｅｔａｄａｔａ™ごとに１つの、４つのセクションの中で記述される。次に以下のこと、すなわち、ＭＤＴにおける時間の表現、ソース・コードのヘッダ・ファイルの中に格納されるべきＭｅｔａｄａｔａ™の種類、およびインストレーション時にＭｅｔａｄａｔａ™のテーブルを埋める問題が記述される。ディメンションはドメインに対する開始のボキャブラリ（ｖｏｃａｂｕｌａｒｙ）である。それらは高レベルのカテゴリーのエンティティを定義する。例えば、小売り業のドメインにおいて、そのディメンションは製品、マーケット、および時間（時間はほとんどのドメインに対して適用できる汎用のドメインである）であり得る。各ディメンションには一組の属性があり、それを使ってそのエンティティを記述することができる。例えば、部門、価格、スタイル、製造者などの、製品に対する各種の属性がある。

【０２６６】このセクションの中のすべてのテーブルはインストレーション時にセットアップされ、あまり変わる可能性はない。というのは、それらはすべてデータ・ウェアハウス２４の構造および業界特有のデータのビューに大きく依存しているからである。このセクションの中のどのテーブルも、一般にエンド・ユーザが変更することはできない。ただし、ＭＤＴアドミニストレータによってＭＤＴのＭｅｔａｄａｔａ™を拡張するため、またはデータ・ウェアハウスの関係スキーマにおける変化に応答してときどき変更が必要となる可能性がある。次の表に示されているように、ディメンションはそれぞれの名前および関連付けられたＩｄによって表される。そのＩｄは他のテーブルとより効率的に結合するために使われる。Ｓｅｇ－ＩｄはこのディメンションのトップレベルのセグメントのＩｄであり、Ｃｏｍｍｅｎｔはコメントである。

【０２６７】

7256 jDim-Id (整数)	名前 (文字列)	Seg-Id (整数)	Comment (コメント) (文字列)
001	製品		
002	マーケット		
…			…

【0268】次の表は各ディメンションに対するすべての属性を表わしている。各属性はユニークなId（Attr－Id）、名前、およびそれらが属しているディメンションのID（Dim－Id）を有する。異なるディメンションに対する属性の名前は同じであってよい（しかし、それらのIDは異なっている）。MDT－Typeはその属性のMDTタイプを示す。各属性は単独のテーブルおよびカラムにマップされ、各属性がマップするデータベースの中のフィールド（Column－Type）の中のフィールドのデータ・タイプを符号化する。Dim－Idフィールドを使ってこのテーブルから与えられたディメンションに対するすべての属性を抽出することができる。MDT－Typeに対するenum値は、列挙型（enum）、整数（int）、浮動小数点数（float）、制限された整数（restricted－int）、制限された浮動小数点数（restricted－float）、文字列（string）である。Column－Typeに対するenum値はすべてそのデータ・ウェアハウスによってサポートされているタイプである。

【0269】

Attr-Id (属性Id) (整数)	Name (名前) (文字列)	Dim-Id (ﾃﾞｨﾒﾝｼｮﾝのId (整数)	MDT-Type (列挙型)	Table (ﾃｰﾌﾞﾙ) (文字列)	Column (ｶﾗﾑ) (文字列)	Column-Type (ｶﾗﾑ･ﾀｲﾌﾟ) (列挙型)	Comment (ｺﾒﾝﾄ) (文字列)
006	製造者	001	enum				
016	サイズ	001	int				
0057	ﾘｰｼﾞｮﾝ	002	enum				
017	部門	001	enum				
0099	サイズ'	002	float				
…	…	…	…				

【0270】列挙型の属性に関して次の表はこれらの属性に対する合法的な値を表わしている。これらの値は「男性用衣類」などのユーザ名および「Dept－017」などの実際のデータ値の名前の両方を含む。これらのテーブルの中の情報はその属性に対する「Select…Distinct…」クエリーによって部分的に作成される。

【0271】

Attr-Id （属性Ｉｄ） （整数）	Display-Name （表示名） （文字列）	Data-Name （データ名） （文字列）
００１７	男性用衣類	Dep1-017
００１７	家庭着	
００１７	ハードウェア	
００５７	南国風	
…	…	…

【０２７２】整数型のタイプの属性の場合、次の表が値の適切な範囲および、これがユーザに対して定義されるのに便利である場合に「代表値」を定義する。その整数属性の範囲および代表値が自然な値でない場合、この表の中にすべての整数属性が現われる必要はない。

【０２７３】

Ａｔｔｒ－Ｉｄ （属性Ｉｄ） （整数）	Ｍｉｎ （最小値） （整数）	Ｍａｘ （最大値） （整数）
「年齢」	０	１２０

【０２７４】浮動小数点数型のタイプの属性の場合、次の表が値の適切な範囲およびこれがユーザに対して定義されるのに便利である場合に「代表値」を定義する。その整数属性の範囲および代表値が自然な値でない場合、この表の中にすべての整数属性が現われる必要はない。

【０２７５】

Ａｔｔｒ－Ｉｄ （属性Ｉｄ） （整数）	Ｍｉｎ （最小値） （浮動小数点数）	Ｍａｘ （最大値） （浮動小数点数）
「収入」	０	１０００．００

【０２７６】ＭＤＴのＭｅｔａｄａｔａ™２５の主要部分は各ディメンションに対する一組のセグメントおよびパーティションである。セグメントは問題のオブジェクトのクラスを定義する一組の属性制限である。例えば、「過去１年以内に改装された店舗」、あるいは「季節的でない店舗全体での販売促進」、あるいは「サイズ

が１４以上であるデザイナーXのシャツ」などである。セグメントは各ディメンションに対して１つの階層の中に配列される。階層はさらにパーティションの概念を使って編成されている。パーティションは単独の属性についての制限によってのみ異なっている一組の関連付けられたセグメントである。セグメントおよびパーティションは各ディメンションに対するセグメント／パーティションの階層を捕捉する一組のテーブルによって表わされ、そして各テーブルに対する属性の制限を定義する。

【０２７７】次の表は各セグメントおよびそれがその一部であるディメンションを示し、そのセグメントの名前および所有者を提供する。グローバルに所有されているセグメントに対する所有者の文字列はヘッダ・ファイルの中で定義される。それはここおよび他の場所で「ＡＬＬ（すべて）」として示されている。各ディメンションに対して名前が「ＡＬＬ　Ｘ（すべてのＸ）」である、いわゆる「トップレベル」のセグメントがある。ここでXはそのディメンションの名前である。Ｎｕｍ－Ａｔｔｒｓフィールドはこのセグメントを制限するために使われる属性の数を含んでいる。「トップレベル・セグメント」では、Ｎｕｍ－Ａｔｔｒｓは０に等しくなる。

【０２７８】

Seg-Id (セグメントId) (整数)	Dim-Id (ディメンションId) (整数)	Name (名前) (文字列)	Owner (所有者) (文字列)	Num-Attrs (属性数) (整数)	Comment (コメント) (文字列)
１１２	００１	男性用衣類	ＡＬＬ		
２６	００１	男性用シャツ	ＡＬＬ		
１４	００１	デザイナーXのシャツ	ｐｇｓ		
１１７	００１	デザイナーXの男性用シャツ	ｐｇｓ		
...	...	...	...		

【０２７９】次の表は次の時間間隔表記で表わされている各セグメントに対するすべての数値属性の制限を表わす。上記の最初の行の中で、Ａｔｔｒ－Ｉｄ　０１７が属性「サイズ」を表わす場合、この行は「０＜＝サイズ＜＝１００」を示すことになる。すなわち、「サイズが０より大きく、そして１００より小さい」である。値は文字列として表わされているので、他のタイプ（浮動小数点数または通貨型など）の属性の制限も表わすことができる。Ｏｐｅｒａｔｏｒ－１（演算子１）のｅｎｕｍ値はｇｒｅａｔｅｒ　ｔｈａｎ（より大きい）、ｌｅｓｓ　ｔｈａｎ（より小さい）、ｇｒｅａｔｅｒ　ｏｒ　ｉｓ（より大きいかまたは等しい）、ｌｅｓｓ　ｔｈａｎ　ｏｒ　ｉｓ（より小さいかまたは等しい）、ｉｓ（等しい）、ｉｓ　ｎｏｔ（等しくない）、ｉｓ　ｂｅｔｗｅｅｎ（間にある）である。

【０２８０】

Seg-Id (ｾｸﾞﾒﾝﾄId) (整数)	Attr-Id 属性Ｉｄ) (整数)	Value-1 (値１) (浮動小数点数)	Operator-1 (enum) (演算子１)	Value-2 (値２) 浮動小数点数)
１１２	０１７	0	「between」	１００
１１２	０１８			
１４				
１１７				
…	…	…	…	…

【０２８１】次の表は各セグメントに対する列挙型の属性の制限のすべての集合を表わしている。「Ｄａｔａ Ｎａｍｅ（データ名）」のカラムは前と同じである。例えば、セグメント（東海岸の都市）を表わす場合、それを集合［ニューヨーク、ボストン、ワシントン…］として定義することができる。このためには、各都市に対して１つづつのいくつかのエンティティーがこのテーブルの中に現われる。このテーブルは文字列の属性制限を表わすためにも使うことができる。Ｏｐｅｒａｔｏｒに対するｅｎｕｍ値は、ｉｓ（である）、ｉｓ　ｎｏｔ（でない）、ｉｓｉｎ　ｌｉｓｔ（リスト内にある）、ｎｏｔ　ｉｎ　ｌｉｓｔ（リスト内にない）である。

【０２８２】

Seg-Id (ｾｸﾞﾒﾝﾄId) (整数)	Attr-Id (属性Ｉｄ) (整数)	Operator (演算子) (ｅｎｕｍ)	Data Name (データ名) (文字列)
１１２	０１９	「in list」	Seattle
１１２			
１４			
１１７			
…	…	…	…

【０２８３】次の表は各パーティションおよびそれについて定義されている属性を定義している。デフォールトのパーティション名は上で定義されている属性のパーティション名と同じである。この場合、ユーザ・インターフェースはプリフィックスとして「ｂｙ－」を付加することによってそのパーティション名を表わす。その名前はユーザ１４０５によって更新される可能性がある。

【０２８４】

Ptrn-Id	Name	Owner	Attr-Id

(ﾊﾟｰﾃｨｼｮﾝId) (整数)	(名前) (文字列)	(所有者) (文字列)	(属性Id) (整数)
59		ALL	0059
117		pgs	0017
…	…	…	…

【0285】次の表およびその次の表はセグメント／パーティションの階層を定義する。この表は各セグメントの子パーティションを表わす。この表は与えられたパーティションに対する親セグメントを見つけるためにも使うことができる。

【0286】

Seg-Id セグメントId) (整数)	Prtn-Id (パーティションId) (整数)
112	59
13	117
…	…

【0287】次の表は各パーティションに対する子セグメントを表わす。この表は各セグメントに対する親のパーティションを見つけるためにも使うことができる。

【0288】

Prtn-Id (パーティションId) (整数)	Seg-Id (セグメントId) (整数)
19	15
221	222
…	…

【0289】測定項目はデータについて測定できるデータ・ウェアハウス24の中の値である。例えば、売上高、価格、および市場占有率はすべて小売業のドメインにおける測定項目である。異なる測定項目は「ロールアップ」される方法が違っている。例えば、いくつかの市場にわたる売上高は加算され、一方、市場占有率は平均化される。本発明のMDTはインストレーション時に一組の基本測定項目を提供し、ユーザはそれらを組み合わせた数式を使って、より複雑な組み合わせの測定項目を形成することができる。次の表は測定項目を示し、このロールアップ機構を定義し、そして測定項目をテーブルおよびカラムにマップする。Display Unit（表示単位）カラムはInfoFrame™の作成のみに対するものである。Precision（精度）は小数点の右側に必要な桁数である。Rollup（ロールアップ）に対するenum値はadd（加算）、av

ｅｒａｇｅ（平均）、ｃｏｕｎｔ（カウント）である。　【０２９０】

BM-Id (整数)	Name (名前)	Rollup (ロールアップ)	Table (テーブル)	Column (カラム)	Display units	Precision	Comment (コメント)
	(文字列)	(enum)	(文字列)	(文字列)	(表示単位) (文字列)	(精度) (整数)	(文字列)
055	売上高	add					
0917	割引率	average					
...	...	...					

【０２９１】基本の測定項目、ある種の二項および単項の演算を組み合わせることによって複合測定項目、およびそれが示す特殊キーワードが作られる。例えば、ターゲット・セグメント、パーティションの子セグメント、あるいは、ターゲット・セグメントの兄弟セグメントが作られる。さらに、次の表を使って、１つの測定項目を制限するために１組のセグメントを符号化することができる。Ｌｅｆｔ－Ａｒｇ（左の引数）およびＲｉｇｈｔ－Ａｒｇ（右の引数）はここに示されているように「Ｌｅｆｔ－Ｍ（左のＭ）」または「Ｒｉｇｈｔ－Ｍ（右のＭ）」の中の引数の種類を符号化する。

１．基本測定項目
２．複合測定項目
３．「ターゲット・セグメント」
４．「親セグメント」
５．セグメント・リスト：２ｄの引数はＳｌｉｓｔのインデックス（次の表参照）である。
６．兄弟セグメント
７．子セグメント

【０２９２】Ｏｐの中の演算子が単項演算子「ｃｏｕｎｔ（カウント）、ｓｕｍ（合計）、ａｖｅｒａｇｅ（平均）」である場合、Ｌｅｆｔ－Ｍだけが使われる。Ｏｐの中の演算子が二項演算子（＋、－など）である場合は、Ｌｅｆｔ－ＭおよびＲｉｇｈｔ－Ｍの両方が使われる。

【０２９３】

CM-Id (整数)	Owner (所有者) (文字列)	Name (名前) (文字列)	Display Unit (表示単位) (文字列)	Left-M (整数)	Left-Arg type (左側の引数のタイプ) (enum)	Op (enum) (整数)	Right-M	Right-Arg type (右側の引数のタイプ) (enum)

【０２９４】複合測定項目がセグメントのリストを指す場合、そのリストの要素は次の表で表わされる。

【０２９５】

CM－Id (整数)	Seg－Id (セグメントId) (整数)
17	5
17	6

【0296】次の表はディメンション的なクエリーを評価するために基本の測定項目と1つの属性を結合するために使われる。その考えは、各属性（1つのテーブルおよび1つのカラムにマップする）および各測定項目（これも1つのテーブルおよび1つのカラムにマップする）に対して、それらの2つのテーブルを結合（ｊｏｉｎ）できる必要があるということである。この表は結合において使うことができる2つの等価カラム（キー）に対するカラム名を示す。

【0297】

Attr-Id (属性Id) (整数)	BM-Id (整数)	Attr-Colom (属性カラム) (文字列)	BM-Column (BMカラム) (文字列)
5	8	「cust_age」	「mkt_share」

【0298】次の表はディメンション的はクエリーを評価するために2つの属性を一緒に結合するために使われる。すなわち、前の表（上記）がその測定項目に対するディメンション的なクエリーの中のすべての属性を結合するのに不十分であった場合、この表をサーチして、この表はすべての属性テーブルをすべての測定項目テーブルと組み合わせるために複数の結合を生成するために使うことができる属性の径路を見つけることができる。

【0299】

Attr1-Id (属性1のId) (整数)	Attr2-Id (属性2のId) (整数)	Attr1-Column (属性1のｶﾗﾑ) (文字列)	Attr2-Column (属性2のｶﾗﾑ) (文字列)
17	4	「cust_age」	「age」

【0300】測定項目間の関係は測定項目間の因果関係の定性的な記述である。例えば、一般に、1つの製品に対する「棚の空間」が増えた場合、「売上高」も同様に増えると予測される。この例では、「棚のスペース」は独立測定項目であり、「売上高」は従属測定項目である。しかし、測定項目間の関係は因果関係および方向の単純な文よりは複雑である。上記の例では「棚のスペース」のすべての値について成立するのではなく、ある範囲の値だけに成立すると期待される。同様に、「棚のスペース」がある程よいパーセントだけ増加した場合にのみ、その関係が成立すると期待することができる。また、測定項目間の関係はあるセグメントに対してのみ成立し、他のセグメントに対しては成立しないと期待される場合もある。

【0301】次の表は基本の測定項目間の関係を次のように定義する。カラムＩ－Ｄｉｒｅｃｔｉｏｎ（Ｉの方向）はヘッダ・ファイルの中に定義されている1つの列挙型の「ｄｉｒｅｃｔ（正方向）」または「ｉｎｖｅｒｓｅ（逆方向）」のいずれかである。その値が「ｄｉｒｅｃｔ」であった場合、その独立測定項目が上昇すると、従属測定項目が上昇する筈である。その値が「ｉｎｖｅｒｓｅ」であった場合、その独立測定項目が上昇す

ると、従属測定項目は下降する答である。Ｉ－Ｄｉｒｅｃｔｉｏｎに対するｅｎｕｍ値はｄｉｒｅｃｔ（正方向）、ｉｎｖｅｒｓｅ（逆方向）である。

【０３０２】

MR-Id (測定項目間の関係のId) (整数)	Owner (所有者) (文字列)	Independent M-Id (独立測定項目のId) (整数)	I-Direction (I-方向) (enum)	Dependent M-Id (従属測定項目のId) (整数)
019	PGS	5	「direct」	21

【０３０３】次の表は測定項目間の関係を独立測定項目の値のある範囲に制限する。演算は＞、＜、＞＝、＜＝、＝、ｎｏｔ＝、またはｂｅｔｗｅｅｎである。ｂｅｔｗｅｅｎの場合、Ｖａｌｕｅｌ（値１）とＶａｌｕｅ２（値２）との両方が使われる。演算に対するｅｎｕｍ値は、ｉｓ　ｌｅｓｓ　ｔｈａｎ（より小さい）、ｉｓ　ｇｒｅａｔｅｒ　ｔｈａｎ（より大きい）、ｉｓ　ｌｅｓｓ　ｔｈａｎ　ｏｒ＝（より小さいかまたは等しい）、ｉｓ　ｇｒｅａｔｅｒ　ｔｈａｎ（より大きいかまたは等しい）、ｉｓ（等しい）、ｉｓ　ｎｏｔ（等しくない）、ｂｅｔｗｅｅｎ（間にある）、ｎｏｔ　ｂｅｔｗｅｅｎ（間にない）である。

【０３０４】

MR-Id (測定項目間の関係のId) (整数)	Operation (演算子) (enum)	Value-1 (値１) (浮動小数点数)	Value-2 (値２) (浮動小数点数)
019	「between」	5	100

【０３０５】次の表は変動分析のアナリスト定義付きの測定項目間の関係に対してのみ適用される。それは変動分析の期間にわたって適切に従属測定項目が変化するときのみ測定項目間の関係が適用されるように制限する。

ｄｉｒｅｃｔｉｏｎ（方向）のｅｎｕｍ値は、ｉｎｃｒｅａｓｅｓ（増加する）、ｄｅｃｒｅａｓｅｓ（減少する）である。

【０３０６】

MR-Id (測定項目間の関係のId) (整数)	derection (方向) (enum)	Value (値) (浮動小数点数)	Unit (単位) (文字列)

【0307】次の表はセグメントごとに測定項目間の関係の適用可能性を制限する。測定項目間の関係が与えられたセグメントに対して適用される場合、それはすべての子セグメントに対しても適用される。

【0308】

MR-Id 測定項目間の関係のId) (整数)	Seg-Id (セグメントId) (整数)
055	19
…	…

【0309】（ａ）時間はＭＤＴのＭｅｔａｄａｔａ ＴＭ２５の１つの重要な部分である。１つのレベルにおいて、ユーザが別のディメンションとして時間のことを考えることが望ましいが、しかし、時間のセグメントはオン・ザ・フライで生成される可能性があるので（例えば、セグメント「年から日付へ」）、それらは異なって表示される。この表示をテーブルの集合として示す。ただし、その情報のいくつかはソース・コードのヘッダ・ファイルの中で定義することができる。次の表はデータ・ウェアハウス２４の中で表現される時間の細かさの最低単位を定義する。ここでデータ・ウェアハウスの中の時間のすべての表現はこの単独の時間の単位を使うことを仮定している。Ｂａｓｅ　Ｕｎｉｔ（基本単位）に対するｅｎｕｍ値は、ｄａｙ（日）、ｗｅｅｋ（週）、ｍｏｎｔｈ（月）、ｙｅａｒ（年）である。

【0310】

Base Unit (基本単位) (enum)
Day

【0311】データベースのテーブルが測定項目によって使われる場合、そのテーブルはその中に時間に対する１つのカラムを備えていなければならない。次の表は各基本測定項目に対する時刻を表わす各テーブルに対するカラムをリストしている。

【0312】

Table (テーブル) (文字列)	BM-Id (基本測定項目のId) (整数)	Column (ｶﾗﾑ) (文字列)
ﾄﾗﾝｻﾞｸｼｮﾝ		ﾀｲﾑ･ｽﾀﾝﾌﾟ

【0313】次の表はユーザ１４０５に対して重要である可能性のある、年の異なる表記を定義する。Ｗｅｅｋ　Ｓｔａｒｔ　Ｄａｙ（週の開始日）に対するｅｎｕｍ値は、Ｓｕｎｄａｙ（日曜日）、Ｍｏｎｄａｙ（月曜日）、Ｔｕｅｓｄａｙ（火曜日）…、Ｓａｔｕｒｄａｙ（土曜日）である。

【0314】

Name （名前） （文字列）	Start Month （開始月） （文字列）	Start Day （開始日） （整数）	Week Start Day （週の開始日） （enum）	Comment （コメント） （文字列）
Calendar （カレンダー）	January （1月）			
Fiscal （会計）	May （5月）			

【0315】

Jan Start 1月 開始 （整数）	Feb Start 2月 開始 （整数）	Mar Start 3月 開始 （整数）	Apr Start 4月 開始 （整数）	May Start 5月 開始 （整数）	Jun Start 6月 開始 （整数）	July Start 7月 開始 （整数）	Aug Start 8月 開始 （整数）	Sept Start 9月 開始 （整数）	Oct Start 10月 開始 （整数）	Nov Start 11月 開始 （整数）	Dec Start 12月 開始 （整数）

【0316】

Quarter 1 Begin Month （第1四半期の開始月） （整数）	Quarter 1 Begin Day （第1四半期の開始日） （整数）	Quarter 2 Begin Month （第2四半期の開始月） （整数）	Quarter 2 Begin Day （第2四半期の開始日） （整数）	Quarter 3 Begin Month （第3四半期の開始月） （整数）	Quarter 3 Begin Day （第3四半期の開始日） （整数）	Quarter 4 Begin Month （第4四半期の開始月） （整数）	Quarter 4 Begin Day （第4四半期の開始日） （整数）

【0317】シリアルＤＡＩ（Ｓ－ＤＡＩ）１４Ａ（図２６）は各クライアント１２に対してＭＤＴのＭｅｔａｄａｔａ™２５に対するアクセスを提供することを担当する。クライアント１２がＭＤＴにログインするとき、シリアルＤＡＩ１４Ａが生成され、そのクライアントのセッションに接続される。更新要求などの、Ｍｅｔａｄａｔａ™に対するすべてのクライアント要求はシリアルのＤＡＩ１４Ａによって処理される。さらに、そのＳ－ＤＡＩ１４ＡはＭＤＴアドミニストレータに対するアクセスを提供し、セットアップ画面を使ってＭＤＴ

を構成できるようにする。

【0318】シリアルDAIのアーキテクチャーが図27に示されている。シリアルDAI14Aはクライアント12からMDTメッセージの形式で要求を受け取る。各MDTメッセージはS－DAI14Aの中にそれ自身の「応答クラス」を持っている。Metadata™のアクセスおよび単純な更新のためのメッセージはDSM16のMetadata™テーブル・インターフェースを通じて処理される（すなわち、Classicのコンポーネント14AAを使わずに－下記に詳細に定義される）。セグメントを追加または削除するための要求はClassicのコンポーネント14AAを使ってその要求を有効化し、そしてその階層を必要に応じて更新する。

【0319】Metadata™のアクセスおよび単純な更新のサービスはClassicのコンポーネント14AAを使わずに、シリアルDAI14Aによってクライアント12に対して提供される。クライアント12が特定のMetadata™を必要とするときは常に、普通はユーザ・インターフェースの中で提示するために、クライアント12は要求メッセージをS－DAI14Aに対して送信する。S－DAI14Aは適切なMetadata™のテーブル25にアクセスして適切なMetadata™を抽出する。次に、それをパッケージ化して1つまたはそれ以上のメッセージの中に入れてクライアント12に対して返す。Metadata™のインターフェースはパブリックのmetadata™およびこの特定のユーザに対するMetadata™に対するアクセスを自動的に提供する。

【0320】データおよび制御の一般的な流れが図28に示されており、以下に説明される。

・「ステップ2101」クライアント12がMetadata™の要求メッセージをS－DAI14AAに対して送信する。このメッセージはそのメッセージ・タイプおよび必要なパラメータを含む。

・ステップ2102」シリアルDAI14Aはそのメッセージ・タイプを調べ、どのMetadata™テーブル25が必要であるかを決定する。

・「ステップ2103」そのパラメータを使って、S－DAIは正しいMetadata™を求めてMetadata™テーブル25を走査する。

・「ステップ2104」シリアルのDAI14Aはそのmetadata™をパッケージ化する。

・「ステップ2105」シリアルのDAI14Aは1つのMetadata™メッセージをクライアント12に対して送り返す。

【0321】可能性は少ないが、エラーのイベントが発生した場合、S－DAI14Aはサーバのエラー・ログの中にそのエラーをログし、クライアント12に対してエラー・コードを返す。各種の重要度の各種のエラーがあり得る。ユーザ・インターフェースを使ってクライアント12から新しい複合測定項目が追加される。そこで任意の構文チェックまたは意味チェックが行われる。複合測定項目に対する構文は上で示されている。新しい複合測定項目が生成されたことをクライアント12が示すと、シリアルDAIはその情報を前記の適切なMetadata™テーブルに対して追加する。

【0322】測定項目ビルダ画面の中で既存の測定項目をプルアップし、それを編集し、そしてSavc（保存）ボタンを押すことによって、既存の複合測定項目を更新することができる。編集された複合測定項目は同じ複合測定項目のIdを使ってMetadata™テーブル－25に対して保存される。クライアント12はユーザ1405に対する警告メッセージを表示してこの測定項目が1つのアナリストの中で使われている場合、それ以降のInfoFrame™の中に示される結果の意味が異なることになる。複合の測定項目がユーザによって所有されている場合、それを削除することができる。複合測定項目およびその式のエントリは前記の該当のテーブルから削除される。警告メッセージがユーザ1405へ提示され、この複合測定項目がどれかのアナリストによって、あるいは他の複合測定項目によって使われている可能性があることを警告し、それ以降のInfoFrame™は正しく作成されず、「アナリスト－時間」のエラーが発生する。

【0323】新しい測定項目間の関係を追加することは、新しい複合測定項目を追加するのと類似している。すべてのチェック（ある場合）はクライアント12の中で発生する。クライアントが、新しい測定項目間の関係が生成されたことを示すと、シリアルDAI14Aはその情報を該当のMetadata™テーブルに追加する。測定項目間の関係を変更することは簡単である。該当のテーブルが更新され、そしてその測定項目間の関係のIdが保存される。測定項目間の関係はそれがユーザ1405によって所有されている場合に削除することができる。その測定項目間の関係およびそのテーブルのエントリは該当のテーブルから削除される。警告メッセージがユーザに対して提示され、この測定項目間の関係がアナリストによって使われている場合、それ以降のInfoFrame™は完了せず、そして「アナリスト・タイム」のエラーが発生することを警告する。

【0324】また、シリアルDAI14AはMDTアドミニストレータからの要求も処理する。これらの要求はMDTをインストールするため、およびパブリックのMetadata™に対する時々の変更を行うために使われる。シリアルDAI14Aはデータ・ウェアハウス24からクライアント12（データ・ウェアハウスのスキーマ情報およびMetadata™の両方を送信している）に対する情報、およびクライアント12からデータ・ウェアハウス24への情報（この場合にはMet

ａｄａｔａ™に対してのみ）の流れを処理しなければならない。シリアルＤＡＩ１４Ａはデータそのものについての何らかの処理を行うこと、すなわち、完全性または例外のチェックを追加して行うことは期待されていない。

【０３２５】データ・ウェアハウスのスキーマは１つの解釈されないハンドルでクライアント１２に対して渡され、クライアント１２がそれを解釈する。ＭＤＴをセットアップあるいはインストールするためのステップが８ステップある。各ステップはクライアント１２がアドミニストレータからの情報をユーザ・インターフェースを通じて受け取り、情報をＳ－ＤＡＩ１４Ａに対して渡すことに関係する。Ｓ－ＤＡＩ１４ＡはＤＳＭインターフェースを通してＭｅｔａｄａｔａ™のテーブルを更新する。これらのステップは次のとおりである。

１．ディメンションを定義する
２．データベースのカラムに対して属性を定義し、そしてマップする
３．符号化された属性値をリネームする
４．基本のセグメントを生成する
５．基本の測定項目をデータベースのカラムに対して定義し、そしてマップする
６．データベース・テーブル間の結合を再チェックする
７．データベースのカラムを時間のマーカとして割り当てる
８．年のタイプを定義する

【０３２６】前に説明されているように、ユーザのセグメントおよびパーティションの階層を表わすためにＣｌａｓｓｉｃのコンポーネント１４ＡＡが使われる、ユーザが１つのセグメントを追加、変更、または削除するとき、Ｃｌａｓｓｉｃのコンポーネント１４ＡＡはそのセグメント／パーティションの階層に対して行われる必要がある任意の、そしてすべての変更を決定し、あるいはいくつかの可能な種類のエラーの１つを検出する。エラーがなかった場合、これらの変更は次にクライアントのインターフェースおよびこの永続的なＭｅｔａｄａｔａ™テーブルを更新するために使われる。これは図２９に示されている。

・「ステップ２２０１」クライアント１２はセグメント・メッセージの追加、削除、または変更のいずれかであるセグメント更新要求を送信する。
・「ステップ２２０２」Ｓ－ＤＡＩはその要求を受信する
・「ステップ２２０３」次にＳ－ＤＡＩはＣｌａｓｓｉｃのモジュールの中の適切な関数を呼出す。その関数はＣｌａｓｓｉｃを使ってその要求によって誘導されるセグメント／パーティションの変更のすべてを決定する。
・「ステップ２００４」これらの変更（または適切なエラー状態）が返される。エラーがあった場合、これはＭｅｔａｄａｔａ™のテーブルを変更せずにクライアントに対して戻される。
・「ステップ２２０５」エラーがなかった場合、すべての変更がＭｅｔａｄａｔａ™のテーブルへ渡される。
・「ステップ２２０６」ふたたび、エラーがなかったと仮定して、アクノレッジ・メッセージがクライアントに対して送られる。
・「ステップ２２０７」知識ベースが変更された場合、該当の知識ベースのファイルが更新される。

【０３２７】ユーザのセグメントの階層はサーバのディスク上の知識ベース・ファイルの中に永続的に保管される。これはそれをＭｅｔａｄａｔａ™のテーブルから常にそれをそれぞれ再生成するには時間が掛かりすぎるからである。各ディメンションおよび各ユーザに対して１つの知識ベース・ファイルがある。各ファイルはサーバのディスク上のあるエリア内にあり、「ｄｉｍｅｎｓｉｏｎ．ｕｓｅｒ」と命名されている。

【０３２８】ユーザ１４０５は既存のセグメント（おそらくトップ・レベルのセグメント）を選択することによって、そして一連の属性および制限事項を選択することによって、新しいセグメントを追加する。このシーケンスはパーティションおよび属性のシーケンスを定義する。［年齢＞６５、収入＞１００Ｋ］の属性および制限事項の集合を考える。このセグメントは「裕福な年配者（Ｗｅａｌｔｈｙ　Ｓｅｎｉｏｒｓ）」と呼ばれる。これはそのセグメントがトップ・レベルにおいて定義されたと仮定して、次のシーケンスを表示させる。

【３０２９】すべての顧客
［Ｂｙ－Ａｇｅ（年齢別）］　＜＝その属性によって命名されたパーティション
年齢＞６５　＜＝自動的に命名された中間セグメント
［Ｂｙ－Ｉｎｃｏｍｅ（収入別）］
裕福な年配者＜＝ユーザ定義のそしてユーザ命名のセグメント

【０３３０】属性の順序は非常に重要である。すなわち、最終セグメント「裕福な年配者」は収入によって、そして次に年齢によって定義され、同じ結果の最終セグメントとなる。しかし、自動的に生成された中間セグメントおよび２つの自動的に生成されたパーティションは異なることになる（この場合、第一のパーティションは「Ｂｙ－Ｉｎｃｏｍｅ（収入別）」であり、中間セグメントは「収入＞１００」となり、そして第２のパーティションは「年齢別」となる。）。

【０３３１】次のガイドラインが新しいセグメントの生成に対してサポートされる。

１．最終のユーザ命名型のセグメントが生成される
２．「サポートしている」パーティションおよびセグメントは自動的に生成されて命名されるが、それはユーザ１４０５によって作成された属性の順序でのみ行われる。
３．　最終セグメントが既存のどれかのパーティション

の子であった場合、それは同様に「そこに現われる」。
４．　最終セグメントが既存のセグメントと単独の属性だけ違っている場合、中間のサポート・パーティションが１つ生成される。
５．　どれかのセグメントが新しいセグメントの下に分類され、そして１つの属性だけが違っている場合、その該当するパーティションが生成される。
上記のガイドライン４を説明するために、顧客の階層が次のようになっていると仮定する。
【０３３２】すべての顧客
［収入別］
中程度の収入
高収入
【０３３３】そのユーザは上記のように「裕福な年配者」を定義する。その階層はここでは次のように見える。
【０３３４】すべての顧客
［年令別］
年令＞６５
［収入別］
裕福な年配者
［収入別］
中程度の収入
高収入
［年令別］
裕福な年配者　＜＝セグメントおよびパーティションが自動的に追加される。
【０３３５】同様に、ユーザ１４０５が新しいセグメントに追加する時、単独の属性によって異なるそのセグメントの下に属している既存のセグメントがある場合、その既存のセグメントは新しいセグメントの下に配置され、そして新しいパーティションが生成される。セグメント／パーティションの階層はやや奇妙なものである。それはそのディメンションに対する「トップレベルのセグメント」に根ざしている。そのディメンションは属性の制限のない１つのセグメントである。それが「ＡＬＬ－Ｘ」を表す理由である。ここでXはディメンションで、属性の制限のないセグメントである。（それが「ＡＬＬ－Ｘ」を表す理由である。ここでXは顧客または製品などのディメンションである）。各セグメントはいくつかの子パーティションを備えている。各パーティションは単独の属性によるセグメンテーションを表し、そしていくつかの子セグメントを持っている。１つのセグメントはいくつかのパーティションに属することができる。１つのパーティションはただ１つの親セグメントを持っている。
【０３３６】ユーザ１４０５は開始セグメントまたは親セグメントを選択し、最終セグメントに対する名前を選定し、そして次に一組の属性および制限事項をそのユーザのインターフェースの中に入力することによって、新しいセグメントを生成する。属性の各制限に対して１つのセグメントが生成され（ユーザが選定した順序で）、セグメント／パーティションの階層に追加される。１つのセグメントが追加される時、パーティションはその新しいセグメントの上および下の両方に生成される必要がある可能性がある。
【０３３７】図３０はセグメントの追加（一般性を失わずに、その親から１つの属性制限だけが違っているもの）の追加を示している。先ず最初に、新しいセグメントはその指定された親の下にあるパーティションに「フィットしなければならない。その新しいセグメントが既存のパーティションまたは複数のパーティションにフィットした場合、それはそれらのパーティションに対して追加される（参照番号２３０１）。フィットしなかった場合、新しいパーティションが生成される（参照番号２３０２）。すべての新しいパーティションのように、他のセグメントはフィットして追加される。その新しいセグメントの他のすべての親（Ｃｌａｓｓｉｃの場合はすべての親）先ず最初にその新しいセグメントと親が１つの属性だけ違っていることを確認する。その場合、参照番号２３０３における親において成立するように、ふたたび既存のパーティション（参照番号２３０４）に対して追加するか、あるいは新しいものを生成する。（参照番号２３０５における親は２つ以上の属性だけ異なる。これは介在しているセグメントの順序を知らないことを意味する。したがって、それを孤立させる。）最後に、その新しいセグメントの子セグメント（参照番号２３０６のような）（Ｃｌａｓｓｉｃの場合はセグメントの子であるという）は新しいパーティション、あるいは参照番号２３０７のような既存のパーティションにフィットされる。
【０３３８】次に、新しいセグメントをユーザの階層に対して追加するためのアルゴリズムについてごく簡単に説明する。Ｃｌａｓｓｉｃ１４ＡＡが使われる時、これは「アンダーラインを付けること」によって示される。
入力：　親セグメントおよび新しいセグメント（ＮＥＷ）。新しいセグメントは単独の属性制限が追加されたものから構成される。ユーザ１４０５が２つ以上の属性制限を入力すると、この基本のアルゴリズムは何回が呼び出され、その中間のセグメントに対して名前が作成される。
【０３３９】ＮＥＷに対して一時的なＣｌａｓｓｉｃのセグメントを生成する。ＮＥＷが既存のセグメントと同一である場合は戻る。

```
／＊ 各親に対して：
     そのレベル差が１であることをチェックする。
     既存のパーティションに対してセグメントを追加するか、あるいは
```

```
        パーティションを生成し、セグメントを追加し、次に追加のセグメント
を追加する。
  */
  「NEWの親セグメントを得る」
  for各親{
    if「レベル差==1」{
      fits_lfag=FALSE;
      forこのセグメントの各子パーティション{
        Ifそのセグメントがこのパーティション内にフィットする}
          fits_flag=TRUE;
          そのパーティションに対してそのセグメントを追加する
        }
      }
      if(!fits_flag){
        その親の下に新しいパーティションを生成する
        そのパーティションに対してそのセグメントを追加する
        for現在の親の他のすべての子{
          if「その子がこの新しいパーティションにフィットする」
          それをそのパーティションに追加する
        }
      }
    }
  }
  /* ここでそのセグメントはそのすべての親に対して追加されている。
     Forその新しいセグメントのすべての子:
     そのレベル差が1であるかどうかチェックする
     それを既存のパーティションに対して追加するか、あるいは
     1つ生成する
  */
  「セグメントNEWの子を得る」
    For各子{
      if「レベル差==1{
        forNEWの各子パーティション{
          if子がパーティションにフィットする{
            それをそのパーティションに追加する
            fits_flag=TRUE
          }
          else{
            NEWの下に新しいパーティションを1つ生成する
            それに対してその子を追加する
          }
        }
      }
```

【0340】セグメントの削除は非常に技巧的となる可能性がある。それは単独のセグメントが削除される時には階層の中に各種の変化が発生する可能性があるためである。設計によって削除されるセグメントの子は他のどのパーティションもそれを参照していない場合にそれ自身が削除される必要がある。

【0341】図31を参照して：

すべての親パーティションについて

［2401］その親／セグメントのリンクを取り除く

［2402］他のセグメントがない場合、そのパーティションを削除する。

その親セグメントのすべての子について

［2404］それらがそのパーティションに現在「フィットしている」かどうかを調べる。

すべての親パーティションに対して
［２４０５］それらが冗長であるかどうか、および「壊されている」可能性があるかどうか調べる。
すべての子パーティションについて
［２４０６］そのセグメント／パーティションのリンクを取り除く
［２４０７］そのパーティションを削除する
すべての子セグメントについて
［２４０８］その子セグメントのリンクを削除する
［２４０９］それらに親パーティションがない場合、削除する
［２４１０］そのセグメントを削除する
【０３４２】最終パーティションの中の明示的なセグメントによって捕捉されないエンティティを表す「その他の」セグメント（ここでは「ＯＳ」と呼ばれている）については以前に説明した。例えば、次のように見える階層において
【０３４３】すべての顧客
［年令別］
年令＞６５
［収入別］
裕福な年配者
【０３４４】パーティション「収入別」の下のＯＳは６５才より上の年齢の人であるが、その人の収入では裕福な生活ができない人達を表すことになる。ＯＳは階層の中で明示的には表されない。これはその定義がどんなセグメントが存在しているかによって変わるためである。例えば、ユーザが「中間クラスの年配者」と呼ばれるセグメントを追加した場合、ＯＳの定義は変化することになる。むしろ、そのＯＳは暗黙になっていて、その属性の制限がその兄弟セグメントの制限を取ってそれらをネゲートすることによって計算することができる。
【０３４５】ユーザ１４０５が最初にログインする時、あるいはセグメントの更新要求が受信された時のいずれかにおいて、Ｃｌａｓｓｉｃの知識ベースは永続的なＭｅｔａｄａｔａ™テーブル２５から初期化されなければならない。各ディメンションおよびそのそれぞれのセグメントおよびパーティションを別々の知識ベースとして扱うことができる。その初期化は次の２つの方法で実行できる。（１）Ｃｌａｓｓｉｃ１４ＡＡに対する直接の呼出しを行うことによって、（２）Ｃｌａｓｓｉｃ１４ＡＡが読むことができるＡＳＣＩＩのフラット・ファイルを生成することによって。前者がおそらくより効率的であるが、デバッグおよびエラー時のキャンセルの場合に有利である可能性がある。Ｃｌａｓｓｉｃの知識ベースを生成するための一般的なステップは前に定義されたテーブルを参照して次のように行なわれる。
【０３４６】・　各ディメンションに対して、ディメンションの定義テーブルから読み出す。
⇒　そのディメンションに対して個別にディメンション・フィラーを定義する。
⇒　各数値属性または文字列属性に対して
◇　その属性の役割を定義する
⇒　列挙型の各属性に対して
◇　その属性の役割を定義する
◇　プリミティブ「フィラー」を定義する
◇　列挙型の各属性の値に対して
＊　個々にその値を定義する
⇒　各セグメント定義に対して
◇　すべての属性制限事項を得る
◇　そのセグメントを定義する
◇　そのセグメントを個々に定義する
⇒　すべてのパーティションに対して
◇　１つの個別パーティションを生成する
⇒　セグメントから子パーティションへのすべてのマッピングに対して
◇　個別のセグメントに対して個別のパーティションを追加することによってマッピングを生成する
⇒　パーティションから子セグメントへのすべてのマッピングに対して
◇　個別のパーティションに対して個別のセグメントを追加することによってマッピングを生成する
【０３４７】ＭＤＴのビジネス・レベルのユーザおよびＭＤＴアドミニストレータ（またはアナリスト・レベルのユーザ）の両方によるＭｅｔａｄａｔａ™の追加、削除、および編集（変更）を含んでいる、Ｍｅｔａｄａｔａ™の修正について以下に説明する。Ｍｅｔａｄａｔａ™を修正することはそれを変更することを意味する。１つの実施形態においては、シリアルＤＡＩ１４Ａおよびクライアント・インターフェースによってサポートされる３種類の変更、すなわち、追加、削除、および編集がある。それにかかわるＭｅｔａｄａｔａ™の種類は通常はセグメント、複合測定項目、および測定項目間の関係を含んでいる。Ｍｅｔａｄａｔａ™を修正するユーザ１４０５の種類は、通常の（あるいは「レベル」ユーザ）、およびＭＤＴアドミニストレータまたはアナリスト・レベルのユーザの２種類があり、その両方共この文章の中ではＭＤＴＡとして参照される。ＭＤＴＡはパブリックなＭｅｔａｄａｔａ™を編集する別のユーザとして考えられるのが普通であるが、必ずしも常にそうではない。ここでは一般にマッピングおよび結合の情報などのＭＤＴの他の種類のＭｅｔａｄａｔａ™を変更するＭＤＴＡについては一般的には考えない。次の表はこれらの組合せを要約している。
【０３４８】

主題 ユーザ MDTA	アクション 追加 削除 編集	オブジェクト セグメント 複合測定項目 測定項目間の関係
MDTA	追加 削除	ディメンション 属性 基本測定項目

【0349】MetadataTMの修正はセグメント・ビルダ、測定項目ビルダ、測定項目間の関係ビルダおよびいくつかのセットアップ画面を含んでいるクライアント・インターフェースを通じて行なわれる。制御の一般的な流れはクライアント12からシリアルDAI（S－DAI）14Aへ、セグメントが関与している場合はClassicのコンポーネント14AAへ、そして次にMetadataTMのテーブル25への流れである。各種のマッピングおよびエラーがユーザ1405に対して適宜定義される。

【0350】MetadataTMの修正は2つの理由のために事項的になる。先ず第1に、MetadataTMの間に依存性が存在する。例えば、複数のセグメントと1つのセグメントを参照する複合測定項目との間の依存性、あるいはMDTAによって管理されている「パブリック」MetadataTMと、各種のユーザのプライベートMetadataTMとの間の依存性である。第2に、スケジュールされているかいないかにかかわらず、定義されているアナリストは修正されたMetadataTMに対して参照する可能性がある。いくつかの重要な問題点が対処されなければならない。例えば、

【0351】1．　欠落しているMetadataTM、すなわち、削除されたMetadataTMに対してアナリストが参照する時にどんなことが起こるか？

2．　MetadataTMが削除され、他のMetadataTMがそれを参照する時にどんなことが起こるか？

3．　MDTAがパブリック・セグメントを変更する時、そのユーザのセグメント階層がどのように更新されるか？

4．　MDTAがディメンション、属性、または基本測定項目を削除する時にどんなことが起こるか？

【0352】対策の範囲において2つの極端な方法を想像することができる。1つは、まったく何もチェックしない（「ユーザまかせの」方式）であり、二つ目はすべての潜在的な問題を捕捉して防止しようとする方式である。1つの好ましい実施形態においては、本発明はスペクトルの「ユーザ任せ」側に近いように設計することができる。ただし、他の適切な設計も実装可能である。このことを前提にして、利用できる時はユーザ1405に警告および情報を与え、可能な限り警告、確認なしでユーザ1405を驚かすようなことは何もしないようにすることが望ましい。

【0353】コンカレントDAI（CDAI）14B（図26）はInfoFrameTMを作成するMDTサブシステムである。それの入力はクライアント12またはスケジューラ・サブシステム18からのInfoFrameTM要求であり、それの出力はユーザのリターン・エリア内のファイルの中に含まれている外部HTMLテキスト・レポートを含んでいる1つのInfoFrameTMである。それがどのように構造化されているか、および結果のレポートのフォーマットを含めて、CDAIサブシステム14Bの動作を以下に説明する。

【0354】CDAI14BはUNIXTM またはNTのサーバ・プラットフォーム上で実行している、UNIXTM またはNTのプロセスである。それは次のようなコマンド行によるディスパッチャ・コンポーネントによって呼び出される。

mdtqueryengine［－c＜config＞］［－e＜errlog＞］

ここでconfig名およびerrlog名はそれぞれ、マスタがそれを呼び出す時にマスタのコマンド行からディスパッチャによって継承された、オプションのコンフィギュレーション・ファイルおよびエラー・ログの名前である。

【0355】CDAIの14B機能の或る特徴はコンフィギュレーション・ファイルで定義されている属性の値によって決定することができる。これらの属性値は関心の度合のヒューリスティック、ローカライゼーション・パラメータなどに対するしきい値を指定する。CDAI14Bはユーザに対するInfoFrameTMおよびMDTアドミニストレータに対するエラー・ログをローカルのテキストの中に発生する。しかし、ユーザのローカル（および言語）はアドミニストレータのそれとは同

じではない可能性がある。この理由で、ＣＤＡＩ１４Ｂはローカライズされたテキストを収集するために２つのメッセージ・カタログを参照することができる。ネイティブのカタログはＩｎｆｏＦｒａｍｅ™に対するローカライズされたテキストに対するソースとして使われる。エラー・カタログはエラー・ログに対するローカライズされたテキストのソースとして使われる。

【０３５６】作成されたＩｎｆｏＦｒａｍｅ™のテキストはローカライズ可能である。測定項目、セグメント、および期間のローカライズされた名前はクライアント１２によって提供されるか、あるいはＭｅｔａｄａｔａから抽出される。他のテキストはＭＤＴのメッセージ・カタログの中にパラメータ化された文字列として保管される。これはネイティブ・カタログとして知られている。ＣＤＡＩ１４Ｂはコンフィギュレーション・ファイルに含まれているＭｓｇＣａｔａｌｏｇＰａｔｈ属性の値、およびクライアントによって提供されているＩＳＯの言語名から、ネイティブ・カタログの名前を計算する。ＭＤＴアドミニストレータの言語でのエラー・ログを作成するために、別のカタログがキープされていることが好ましい。

【０３５７】ＣＤＡＩ１４Ｂは１つのｍｄｔ＿ＩｎｆｏＦｒａｍｅＲｅｑｕｅｓｔオブジェクトをＲＵＮ＿ＡＮＡＬＹＳＴ＿ＲＥＱＵＥＳＴメッセージの中に受け入れる。その要求は作成されるべきＩｎｆｏＦｒａｍｅ™を記述している、１ｎｆｏＦｒａｍｅＤｅｆｉｎｉｔｉｏｎ（ｍｄｔ＿ＩｎｆｏＦｒａｍｅＤｅｆｉｎｉｔｉｏｎ）を含むことができる。それはＩｎｆｏＦｒａｍｅ™のタイプ、レポートされるべきセグメント、レポートされるべき測定項目、およびレポート対象の期間を指定することができる。１つの実施形態においては、次の５種類のレポートがある。

【０３５８】・「要約化（Ｓｕｍｍａｒｉｚａｔｉｏｎ）」－１つのターゲット・セグメント上のターゲット測定項目の基本分析

・「変動分析（Ｃｈａｎｇｅ　Ａｎａｌｙｓｉｓ）」－２つの別々の期間についての、および１つのターゲット・セグメントについてのターゲット測定項目の違いの要約化

・測定項目比較（Ｍｅａｓｕｒｅ　Ｃｏｍｐａｒｉｓｏｎ）」－１つのターゲット・セグメントについての２つの期間の間の差の要約化

・セグメント比較（Ｓｅｇｍｅｎｔ　Ｃｏｍｐａｒｉｓｏｎ）」－２つの別々のターゲット・セグメント上での同じ測定項目の間の差の要約化

・「トレンド分析（Ｔｒｅｎｄ　Ａｎａｌｙｓｉｓ）」－ターゲット測定項目と関連の測定項目における、ターゲット・セグメントと関連のセグメントについての、時間軸上での傾向のレポート

【０３５９】その要求はＩｎｆｏＦｒａｍｅ™Ｔｒｉｇｇｅｒ（ｍｄｔ＿ＩｎｆｏＦｒａｍｅＴｒｉｇｇｅｒ）を含むことができる。そのトリガはデータ・ウェアハウス２４およびＡｌｅｒｔフラグの中のいくつかの条件に対するテストを定義する。そして、「ネストされた」　ＩｎｆｏＦｒａｍｅ™要求を含むことができる。その要求がトリガを含んでいる場合、ＣＤＡＩ１４Ｂはその条件が「真」であればＩｎｆｏＦｒａｍｅ™だけを実装しなければならない。その条件が「真」である場合、ＣＤＡＩ１４Ｂは、Ａｌｅｒｔフラグが「真」であれば、アラートを発生し、トリガがネストされた要求を含んでいる場合は、そのネストされた要求を実行する。一般に、ＣＤＡＩ１４Ｂの出力はローカライズされて拡張されたＨＴＭＬのレポートを含んでいるＩｎｆｏＦｒａｍｅ™オブジェクトとなる（図１９参照）。そのＩｎｆｏＦｒａｍｅ™はユーザ・リターン・エリア（下で定義される）の中のファイルに書き込まれる。

【０３６０】ユーザ・リターン・エリアはコンフィギュレーション・ファイルのＵｓｅｒＲｅｔｕｒｎＡｒｅａＰａｔｈパラメータによってパスが与えられるディレクトリである。ＩｎｆｏＦｒａｍｅ™が正しく完了すると、そのレポートはＩＮＦ．＜ＵＩＤ＞．＜ＵＮＱ＞と名付けられる。ここでＵＩＤはユーザＩＤであり、ＵＮＱはこのファイル名がこのユーザに対して作成された他のすべてのファイル名とは異なっている（ユニークである）ことを保証する何らかの文字列である。

【０３６１】ユニークな識別子はディスパッチャがＣＤＡＩ１４Ｂを立ち上げる時にディスパッチャによって発生され、そしてＩｎｆｏＦｒａｍｅ™要求を伴うＣＤＡＩ１４Ｂがそれを利用できるようにする。レポートごとにユニークな名前が必要なので、ＣＤＡＩ１４Ｂのインスタンスはレポートを１つだけしか発生することができない。ＩｎｆｏＦｒａｍｅ™要求が２つ以上のレポートを指定した場合、トリガされた要求がアラート・レポートおよび「その」レポート以外に「その他の」レポートを必要とする時、その要求を受け付けているＣＤＡＩ１４Ｂはこの多重のレポートを処理するために新しいＣＤＡＩ１４Ｂをディスパッチする必要がある。

【０３６２】これらの拡張はクライアント・ビューワによって解釈される。そのレポートはＩｎｆｏＦｒａｍｅ™レポート、アラート・レポートまたはエラー・レポートのいずれかを含む。ＩｎｆｏＦｒａｍｅ™レポートはＣＤＡＩ１４ＢがＩｎｆｏＦｒａｍｅの定義を正しく完成した時に作成される。それはｉｆｇｎ＿Ｒｅｐｏｒｔクラスによって作成される。アラート・レポートはＩｎｆｏＦｒａｍｅ™要求がトリガを有していて、そのトリガが「真」に評価され、Ａｌｅｒｔフラグがセットされている時に作成される。それはｉｆｇｎ＿ＡｌｅｒｔＦｒａｍｅクラスによって作成される。エラー・フレームはＣＤＡＩ１４ＢがトリガまたはＩｎｆｏＦｒａｍｅ™の定義を評価できず、それについて知らせるた

めにある時に作成される。それはｉｆｇｎ＿Ｅｒｒｏｒ Ｆｒａｍｅクラスによって作成される。

【０３６３】ＩｎｆｏＦｒａｍｅ™レポート（図１９参照）の内容は、要求される分析のタイプによってだけでなく、対象となった測定項目によっても大きく変化する可能性がある。そのレポートは例えば、ヘッディング、４つの黒丸付きのパラグラフ、テーブルなど、各種の方法で編成することができる。

「ヘッディング」－アナリストが定義されてそのセグメント、測定項目および分析される期間を命名する時、そしてそのアナリストがトリガ、そのトリガの条件およびそのトリガ条件が満足された時刻を持っていた場合、ユーザによって提供されたアナリスト記述を引用する。

【０３６４】・「ターゲット・セグメント・レポート」－次のものを含む黒丸印付きのパラグラフ

「ターゲット・セグメント」－アナリストの定義の中で直接指定されているセグメントと期間についての測定項目に対する結果を強調表示しているテキスト。追加のテキストはこのセクションではレポートされない。

「親の貢献」－ターゲット・セグメントによってその親セグメントに対してなされた重要な貢献を強調表示している黒丸印付きのパラグラフ。そのターゲット・セグメントがトップ・レベルのセグメントである場合、またはそのターゲット測定項目が１つの親セグメントに対する参照を含んでいる時はこのセクションは含まれない。

「兄弟比較」－ターゲット・セグメントの値のその兄弟セグメントの値に対する興味深い比較を提供している、黒丸印付きのパラグラフ。そのターゲット・セグメントがトップ・レベルのセグメントである時は、このセクションは含まれない。

「兄弟グラフ」－兄弟セグメントの値を示しているバー・チャートまたはパイ・チャート、あるいは傾向を示している線グラフ。ドリル・ダウン・セグメントがある場合は、このグラフは作られない。

【０３６５】・「ドリル・ダウン」－ドリル・ダウン・パーティションの中の子セグメントの値とターゲット・セグメントに対する値との間の興味深い関係を強調表示している黒丸印付きのパラグラフ。ドリル・ダウン・パーティションはユーザがアナリスト定義の中で指定することができる。ユーザがドリル・ダウン・パーティションを何も指定しなかった場合、ＭＤＴは１つまたはそれ以上の興味深いパーティションを自動的に選定する。それらがどのように自動的に選定されるかを知るには、ドリル・ダウン・パーティションの選定についての下記のセクションを参照されたい。子パーティションが存在せず、そして生成する制限の付いていない属性が存在しない時、あるいは興味深い既存の、または生成されたパーティションがない場合、このセクションは含まれない。

【０３６６】「ドリル・ダウン・グラフ」－ドリル・ダウン・セグメントに対する値を示しているバー・チャートまたはパイ・チャート、あるいは傾向を示している線グラフ。興味あるドリル・ダウン・セグメントがない場合、このグラフは作られない。

・「式の分解」－そのコンポーネント測定項目の複合測定項目に対する興味深い貢献を強調表示している黒丸印付きのパラグラフ。ターゲット測定項目が複合測定項目である時だけ、このセクションが含まれる。

・「測定項目間の関係」－ターゲット測定項目の２つの値の間の差（あるいは、測定項目比較分析のためのターゲットと比較測定項目との間の差）のあり得る原因を記述している黒丸印付きのパラグラフ。要約化分析は測定項目間の関係のグラフを含んでいない。

【０３６７】「テーブル」－その分析の間にレポートされたすべての測定項目値を示しているテーブル。親、ターゲットまたは比較のセグメント以外のセグメントはハイパーリンクとしてマークすることができる（ふたたび、例えば、図１９参照）。下線付きのＨＲＥＦはターゲット・セグメントとしてのこのセグメントを同じタイプの新しいアナリストに対して、同じ測定項目、比較、および期間に対して置き換えるために必要な情報を含む。

【０３６８】アラート・レポートはずっと簡単である。その最も興味深い特性は、アナリスト名に対するハイパーリンクである。ＩｎｆｏＦｒａｍｅ™要求が定義された時、そのイベントについてのより多くの情報をオプションとして集めるために１つのアナリストを定義しておくことができる。ハイパーリンクを選択することによって、ユーザはこのアナリストを立ち上げることができる。アラート・フレームのテキストは、例えば、次のように見える。

【０３６９】

```
Ａｌｅｒｔ：<アナリスト名>
<ターゲット・セグメント>
<追加のセグメント１>
．．．
<追加のセグメントｎ>
<基本の時間間隔の記述>
アラートは<アラートがトリガされる時刻>においてトリガされた
いま<アナリスト名>を実行させるにはここをクリックしてください。
トリガは、　<測定項目名>［<演算子><測定項目名>］，
　　<測定項目名>［<演算子><測定項目名>］，
　　．．．
　　<測定項目名>［<演算子><測定項目名>］，
```

【０３７０】エラー・レポートはより単純である可能性がある。それはユーザ１４０５に対して正確に１つの文（発生したエラーの説明）を通信することを意味する。そのフォーマットは次のようになる。

【０３７１】

```
エラー：<アナリスト名>
エラーが<エラー発生時刻>において発生した
<エラーのテキスト>
```

【０３７２】ＣＤＡＩ１４Ｂはエラーをサーバのエラー・ログに対してレポートすることができる。そのエラー・メッセージのテキストはパラメータ化されたローカライズ可能な文字列として、エラー・メッセージのカタログの中に維持される。

【０３７３】４．　ＤＳＭサブシステム１６およびスケジューラ・サブシステム１．８

データおよびスキーマ操作サブシステム１６およびスケジューラ・サブシステム１８が以下に詳細に説明される。ＭＤＴサーバ３２はユーザ１４０５に対して次の２つのクラスの要求を実装することができる。

【０３７４】・対話型要求；ＭｅｔａｄａｔａＴＭのフェッチ、ＭｅｔａｄａｔａＴＭの更新、ＩｎｆｏＦｒａｍｅＴＭのスケジューリング、ＩｎｆｏＦｒａｍｅＴＭのステータスなど。サーバは各クライアントに対して一度に一つの対話型要求を実装する。一度に一つの対話型要求を実装することは、それらが対話型であるのとほとんど同じくらい興味深いので、これらをシリアル要求とも呼ぶ。

・バッチ要求；トリガ要求およびＩｎｆｏＦｒａｍｅＴＭの作成。サーバは複数のバッチ要求を一度に実装することができなければならない。複数のバッチ要求を一度に実装することができなければならないということは、それらがバッチであるということとほとんど同様に興味深いので、これらをコンカレント要求とも呼ぶ。

【０３７５】各コンカレント要求はデータベースに対して１つまたはそれ以上のクエリーを生じさせる。ＯＤＢＣ標準はデータベースに対する非同期のクエリーをサポートするが、ＯＤＢＣの実装の多くはサポートしない。これらの実装において、各コンカレント要求はそれ自身のプロセスを必要とする。各要求をそれ自身のプロセスの中に置くことによって、より多くのＯＤＢＣ実装で作業することができ、複数の要求に対してメモリおよびデータ構造を管理する責任をオペレーティング・システムに負わせ、各コンカレント要求がそれ自身のプロセスを得ることになる。このプロセスはコンカレント・インスタンスとなる。

【０３７６】シリアルの要求および戻りの結果が非同期のイベントであるようにされる場合、単独のサーバ・インスタンスがそのサーバのシリアル要求のすべてを処理することができる。しかし、非同期的に実装することは、難し過ぎることはないにしても、各クライアント１２が単純に新しいサーバのインスタンスを渡されることができる時は、むしろ無意味である。したがって、各クライアント１２にはシリアルのインスタンスが割り当てられ、その対話型の要求を処理する。

【0377】Ｗｉｎｄｏｗｓ　ＮＴ™がスレッドを実装すること、そして将来のＵＮＩＸ™の将来の改訂版のオペレーティング・システムも同様にそのようになることに留意することが重要である。スレッドはオペレーティング・システムに対して非同期のイベントを処理するための責任を渡すための機会を提供し、一方、単独プロセスの中のメモリを依然として管理している。また、本発明の説明の目的のために、シリアルおよびコンカレント・のインスタンスはその要求に対して適切な、完全なサーバ機能のサブセットをそれぞれ実装しているユニークな実行可能プログラムによって実装されることが仮定される。

【0378】図３２を参照すると、サーバ３２は協働しているプロセスの集合として実装される。以下に説明されるように、５つのクラスのプロセスがある。

・マスタ・プロセス２５１１はサーバとのクライアントのコネクションを受け入れることを担当し、そしてクライアント１２にシリアル・インスタンスを割り当てる。

・シリアル・インスタンス２５１２はこのクライアントのシリアル要求のすべてを実行する責任がある。それらはＭｅｔａｄａｔａ™のフェッチおよび更新、ＩｎｆｏＦｒａｍｅ™のスケジューリング要求などである。シリアル・インスタンスはスケジュールされたＩｎｆｏＦｒａｍｅ™の要求をスケジューラのキューの中に待ち行列として入れる。シリアル・インスタンスはクライアントのＩｎｆｏＦｒａｍｅ™作成要求、コンカレント要求も取得するが、それはこれをディスパッチャ上に渡す。

・ディスパッチャ・プロセス２５１３はシリアル・インスタンスまたはスケジューラによってそれに対して渡された各コンカレント要求に対するコンカレント・インスタンスを割り当てること、およびペンディングおよび実行中のコンカレント要求の状態を報告することを担当する。

・コンカレント・インスタンス２５１４は単独のコンカレント要求の実行を担当する。

・スケジューラ・プロセス１８はスケジュールされた時刻においてディスパッチャに対してＩｎｆｏＦｒａｍｅ™要求を渡すことを担当する。

【0379】関係の生成は親から子への白いポインタ２５０１で示されている。要求の関係は送信側から受信側への黒の矢印２５０２によって示されている。サーバのプロセスは単独の状態をクライアントに対して提示するためにいくつかのリソースを共有する。それらはＭｅｔａｄａｔａ™、スケジューラのキューおよびリターン・エリアである。ユーザ１４０５がユーザのＭｅｔａｄａｔａ™に対する変更を行う時、その変更はそのユーザのそれ以降のＩｎｆｏＦｒａｍｅ™要求のすべてに対して見えなければならない。ユーザ１４０５がＭＤＴＡ（アドミニストレータ）であって、そのＭＤＴＡがグローバルのＭｅｔａｄａｔａ™に対する変更を指定する時、それらの変更はそれ以降のすべての要求に対して見えなければならない。

【0380】ＭＤＴのＭｅｔａｄａｔａ™は顧客のデータ・ウェアハウス２４上に格納される。このＭｅｔａｄａｔａ™へのアクセスはＤＳＭ１６によって管理される。そのＭｅｔａｄａｔａ™に対するアクセスを最適化するために、ＤＳＭ１６は共有メモリ、ＭＤＴテーブルのライト・スルー・キャッシュを実装する。各ユーザ１４０５はＭｅｔａｄａｔａ™のサブセットだけを使い、そして実際的な理由のために、そのデータはデータベースのフラットなテーブルからアプリケーション固有の構造に再編成される。そのユーザのシリアルおよびコンカレント・インスタンス２５１２、２５１４の各々がこのサブセットのコピーをキープする必要がある。このイメージはＤＡＩ１４によって構築されて維持される。

【0381】ユーザ１４０５がＭｅｔａｄａｔａ™のアイテムを変更する時、ＤＡＩ１４はその変更をローカル・イメージに対して有効にし、そしてＤＳＭ１６がデータ・ウェアハウス２４を更新するよう指令する。ＤＳＭ１６はこの変更を共有メモリのキャッシュおよびウェアハウス２４に対してライト・スルー・キャッシュの中に書き込む。Ｍｅｔａｄａｔａ™についての関係および操作が図３３の中に概略的に示されている。細い線の矢印２６０１は共有されているメモリのＭｅｔａｄａｔａ™キャッシュ２６１０に対する付加を表している。太い線の矢印２６０２はＭｅｔａｄａｔａ™の径路を表している。

【0382】本発明のＭＤＴは或る将来の時刻で実行されるか、あるいは定期的に実行されるようにスケジュールされているＩｎｆｏＦｒａｍｅ™の要求のすべてをリストする単独のスケジューラ・キューを維持する。ユーザ１４０５がクライアント１２を通してＩｎｆｏＦｒａｍｅ™要求をスケジュールする時、そのスケジュール要求はシリアル・インスタンス２５１２を通してスケジューラ１８に対して渡される。スケジューラ１８はその要求を受け付け、その要求をスケジュール・キューの中に置く。ユーザ１４０５がスケジュールされた要求を削除すると、スケジューラ１８はその要求をスケジューラのキューから削除する。また、ＤＡＩ１４はユーザのスケジュール・ステータス要求も受け付ける。それはスケジューラのキューの検査によってそれらを満たす。

【0383】規則的な時間間隔で、スケジューラ１８はスケジューラのキューを検査し、時期が来ている要求を識別し、それらをディスパッチャ２５１３に対して渡す。スケジューラ１８は次に非繰返し要求をスケジューラのキューから取り除く。ディスパッチャ２５１３はその要求を実行するためのコンカレント・インスタンス２５１４を生成する。コンカレント・インスタンス２５１

4は1つのプロセスであり、プロセスがディスパッチャ２５１３によって管理されなければならない制限されたリソースである筈であると期待する。このように、ディスパッチャ２５１３に対して渡されたＩｎｆｏＦｒａｍｅ™の要求は2つの状態のうちの1つの状態があり得る。それらは（１）「ペンディング（プロセスに対して待機している）」および（２）「実行中」である。ディスパッチャ２５１３はペンディングおよび実行中の要求のリストをキープする。ユーザ１４０５があデータ・ステータス要求を行うと、クライアント１２はそれをシリアルのインスタンス２５１２に対して渡し、ＤＡＩ１４がそれを実装する。それはそのステータス要求を完了するために、ディスパッチャ２５１３からユーザ要求のステータスのリストを収集する必要がある。

【０３８４】スケジューラのキュー２７１０上での関係および動作が図３４の中に概略的に示されている。ＩｎｆｏＦｒａｍｅ™の要求の径路が黒の矢印２７０１によって表されている。スケジュールされた要求およびペンディングおよび実行中の要求に関するステータス情報（それはクライアントのステータス要求に応答してシリアルのインスタンスによって収集されなければならない）は白の矢印２７０２によって表されている。

【０３８５】完成されたＩｎｆｏＦｒａｍｅ™はそれらが完成された時と、クライアント１２がそれらを収集するために呼び出す時との間、リターン・エリア内に駐在している。リターン・エリアはファイル・システム上の単なるディレクトリである。それは各ユーザ１４０５に対するサブディレクトリを含む。コンカレント・インスタンス２５１４が1つのＩｎｆｏＦｒａｍｅ™要求を完了すると、それはそのＩｎｆｏＦｒａｍｅ™をリターン・エリアのユーザのサブディレクトリの中にコピーする。再帰型の要求はリターン・エリア内に多くのＩｎｆｏＦｒａｍｅ™を残す可能性がある。コンカレント・インスタンス２５１４は各レポートに対してユニークな名前を作成しなければならない。

【０３８６】クライアント１２は時によってＩｎｆｏＦｒａｍｅ™のステータス要求をシリアル・インスタンス２５１２に対して渡す。ＤＡＩ１４はその要求を実装する。ＤＡＩ１４はその完了されているＩｎｆｏＦｒａｍｅ™を識別するために、リターン・エリアのユーザのサブディレクトリを検査する必要がある。ＤＡＩ１４はそのファイルメモリを「デコード」して、アナリストの名前および実行日付を作成してクライアント１２に対してレポートすることができる。また、クライアント１２は時によってＩｎｆｏＦｒａｍｅ™の更新要求をシリアル・インスタンス２５１２に対して渡す。ＤＡＩ１４はその要求を実装する。ＤＡＩ１４はリターン・エリアのユーザのサブディレクトリからそのＩｎｆｏＦｒａｍｅ™を収集する。それはそのＩｎｆｏＦｒａｍｅ™をクライアント１２に対して渡し、そしてクライアント１２がその受領をアクノレッジした時、リターン・エリアからそのイメージを取り除く。

【０３８７】リターン・エリア２８１０上の関係および動作が図３５の中に概略的に示されている。完成したＩｎｆｏＦｒａｍｅ™のフローが黒い矢印２８０１によって示されている。クライアントのステータス要求を満足するためにシリアル・インスタンス２５１２によって要求されるステータスの移動は、白の矢印２８０２によって示されている。

【０３８８】１．「ＤＳＭサブシステム１６」

ＤＳＭサブシステム１６について以下に詳しく説明される。ＳＱＬゼネレータはＩｎｆｏＦｒａｍｅ™からのディメンション的クエリを受信し、必要なデータベース・クエリを作成し、そしてその結果をＩｎｆｏＦｒａｍｅ™ゼネレータへ返す。データベースに対するインターフェースはＯＤＢＣを通して行なわれ、ＯＤＢＣはクエリをＳＱＬ文字列の形式にする。ＳＱＬゼネレータはデータベース２４に問い合わせて各測定項目／セグメントのペアを評価しなければならない。その測定項目はｄａｉ＿ＭｅｅａｓｕｒｅＬｉｓｔからのものであり得る。ＱｕｅｒｙＤａｔａｂａｓｅ（）の第1の形式において、そのセグメントはｔａｒｇｅｔＰａｒｔｉｔｉｏｎの子セグメントであり、第2の形式においては、各測定項目は暗黙に決められたターゲット・セグメントに対してのみ評価される。各測定項目は複合測定項目である可能性があり、それはその複合測定項目を構成する測定項目を評価するために複数のクエリを必要とする可能性がある。標準のＳＱＬプログラミング技法を使ってＤＳＭサブシステム１６を実装することができる。

【０３８９】２．「スケジューラ・サブシステム１８」

スケジューラ・サブシステム１８について以下に詳細に説明される。スケジューラ・サブシステム１８は実行されるスケジュール／またはトリガを伴うアナリストをサブミットすることを担当する。また、それはスケジュールされたおよび／またはトリガされたアナリストのリストを維持することも担当し、それらのリストに対する削除、変更および追加も担当する。このセクションでは、スケジュール、トリガ、またはスケジュールとトリガを有するアナリストはその3つのタイプのアナリストの挙動の方法に差がない限り、「スケジュールされた」アナリストと呼ばれる。

【０３９０】ＩｎｆｏＦｒａｍｅ™の要求がＳ－ＤＡＩサブシステム１４Ａによって受信されると、それはそれがスケジュールされたアナリストに関連付けられている場合は、スケジューラ・サブシステム１８に対して渡される。スケジューラ１８はそのアナリストをディスパッチする適切な期間を決定する。それが実行されるようにスケジュールされると、Ｓ－ＤＡＩサブシステム１４Ａがスケジュールされていない要求を渡すのと同じ方法で、ディスパッチャ２５１３に対して1つの要求が渡さ

れる。

【０３９１】図３６はこのプロセスのさらに詳細を示している。Ｓ－ＤＡＩ１４Ａからスケジューラ１８はスケジュールされたＩｎｆｏＦｒａｍｅ™要求を受信し、ユーザ要求を削除およびディスエーブルし、スケジュールされたアナリスト要求を削除する。要求をユニークに識別するために、Ｓ－ＤＡＩ１４Ａはアナリストｉｄ（ＩｎｆｏＦｒａｍｅ™要求オブジェクトの中に含まれている）およびユーザｉｄを提供しなければならない。

【０３９２】スケジュールされたＩｎｆｏＦｒａｍｅ™要求が受信されると、そのアナリストはトリガを有している場合はトリガ・リスト２９０１の中に入れられ、あるいはそれがスケジュールまたはスケジュールとトリガのいずれかを有している場合はスケジュール・リスト２９０２の中に入れられる。トリガされたアナリストと、スケジュールされていてトリガされたアナリストとの間の違いは前者が各トリガ期間において実行されるのに対し、後者はスケジュールされた時のみ実行されることである。スケジューラ１８は２つの実行期間を持っている。１つはトリガされた要求に対するものであり、もう１つはスケジュールされた要求に対するものである。この２つの期間は各ＭＤＴサーバ３２において構成可能であり、ＭＤＴアドミニストレータが変更することができる。

【０３９３】トリガの期間が発生すると、スケジューラ１８はトリガ型のイベントのリスト２９０１を縦覧する。そのタイム・スライスの間に実行するようにスケジュールされているイベントでそのユーザ・アカウントがイネーブルされているものの場合、トリガなしのＩｎｆｏＦｒａｍｅ™要求のコピーがディスパッチャ２５１３へ渡される。スケジュールの期間およびスケジュールのリスト２９０２に対して同様なプロセスが行なわれる。ユーザ１４０５が削除された場合、スケジューラ１８はその削除されたユーザによって所有されているリストからすべてのアナリストを取り除く。ユーザ１４０５がディスエーブルされていた場合、そのリストの中のどのアナリストも、ユーザ１４０５がふたたびイネーブルされるまでは実行されない。アナリストが変更された場合、ユーザ１４０５はすべての関連付けられたスケジュール型要求をイメージ的に取り除かなければならない。さもなければ、それらの古いアナリスト定義によって実行され続ける。

【図面の簡単な説明】

【図１】本発明のシステムのブロック図である。

【図２】図１のシステムの内部のクライアント・サブシステムのブロック図である。

【図３】図１のシステムの内部のデータ抽象化インテリジェンス・サブシステムのブロック図である。

【図４】図１のシステムの内部のデータおよびスキーマ操作サブシステムのブロック図である。

【図５】図１のシステムの内部のスケジューラ・サブシステムのブロック図である。

【図６】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図７】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図８】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図９】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１０】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１１】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１２】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１３】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１４】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１５】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１６】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１７】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１８】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

【図１９】グラフィック・ユーザ・インターフェースを採用しているレポートを生成するためのツールのビューである。

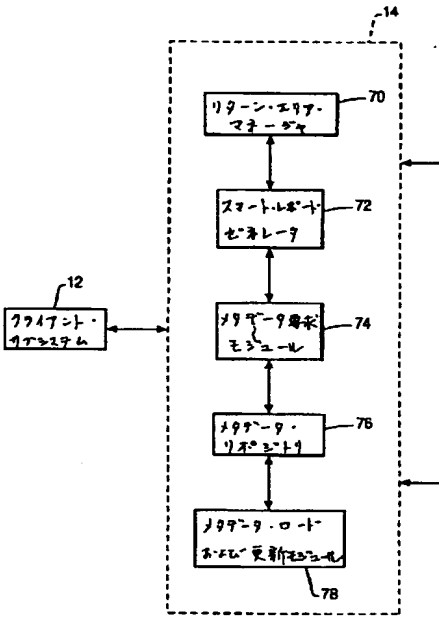
【図２０】Ｍｅｔａｄａｔａ™が生成される方法を説明しているフロー・ダイアグラムである。

【図２１】図１のシステムの内部のクライアント・サブシステムの他のブロック図である。

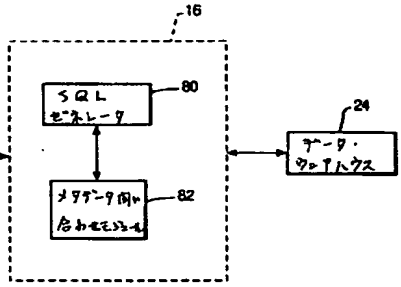
【図２２】図１のシステムの内部のクライアント・サブ

システムのさらに他のブロック図である。
【図２３】本発明の内容にしたがって生成され得るデータベースの階層である。
【図２４】本発明の内容にしたがって生成され得る別の階層である。
【図２５】本発明の内容にしたがって生成され得るさらに別の階層である。
【図２６】ＤＡＩおよび他のサブシステムと本発明のシステムのコンポーネントとの間の一般的な高レベルのデータ・フローである。
【図２７】図１のシリアルＤＡＩサブシステムのアーキテクチャである。
【図２８】図１のシリアルＤＡＩシステムのフロー・ダイアグラムである。
【図２９】図１のシリアル・ＤＡＩシステムに対する別のフロー・ダイアグラムである。
【図３０】方法のシステムにおけるデータ・セグメントの追加を示す図である。
【図３１】本発明のシステムにおけるデータベース・システムの削除について示す図である。
【図３２】図１のサーバによって実行されるプロセスを示す図である。
【図３３】Ｍｅｔａｄａｔａ™に関する関係および動作を示す図である。
【図３４】スケジューラのキューについての関係および動作を示す図である。
【図３５】リターン・エリアについての関係および動作を示す図である。
【図３６】アナリストによって実行される要求を示す図である。

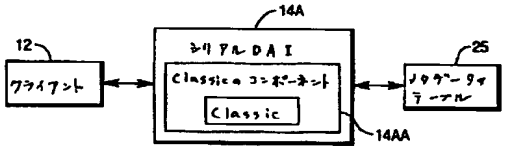
【図３】



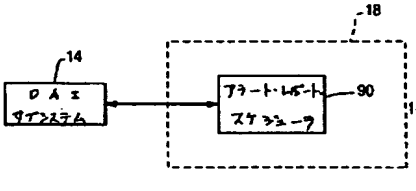
【図４】



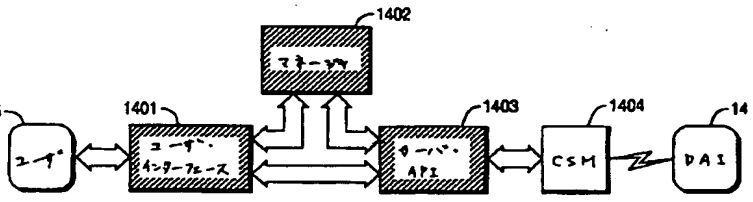
【図２７】



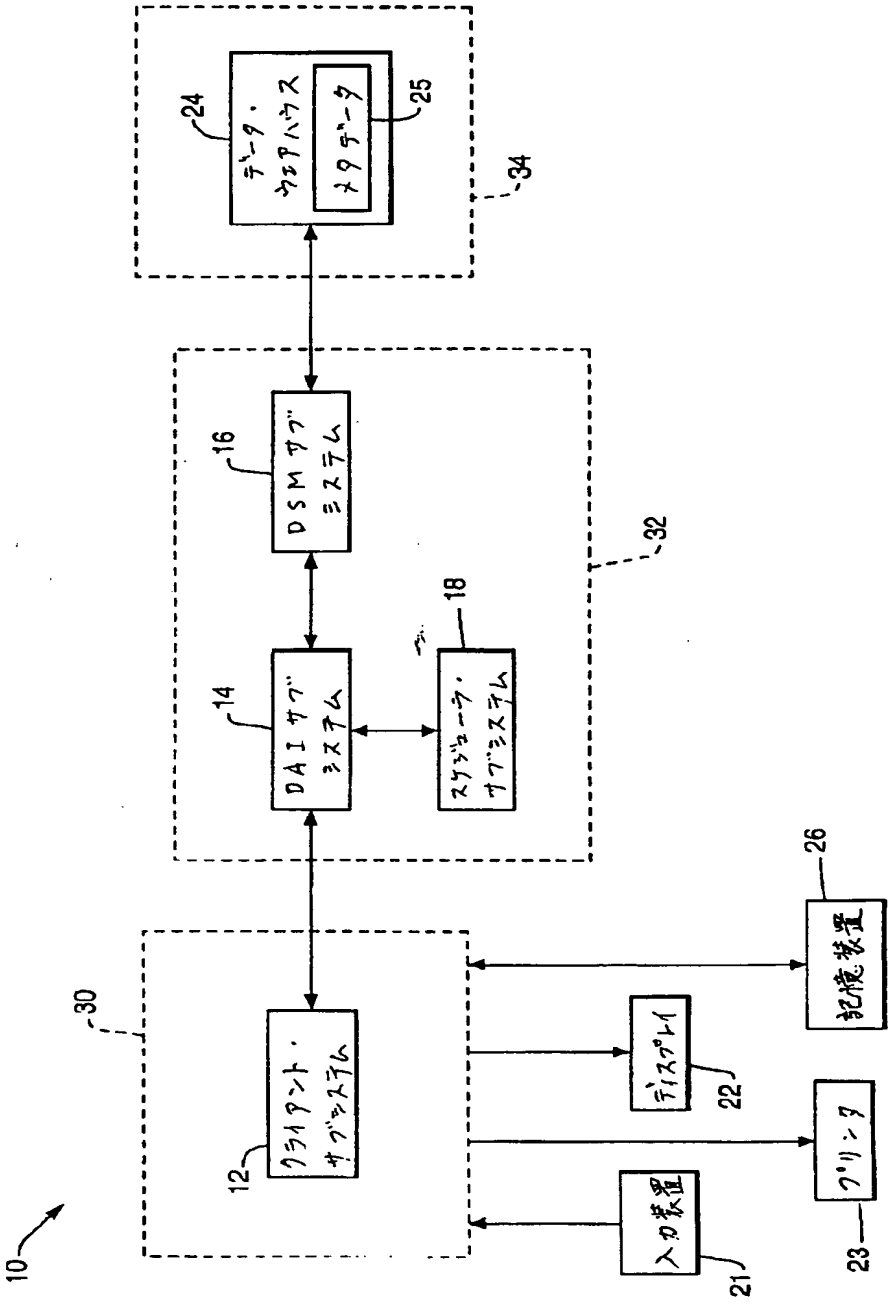
【図５】



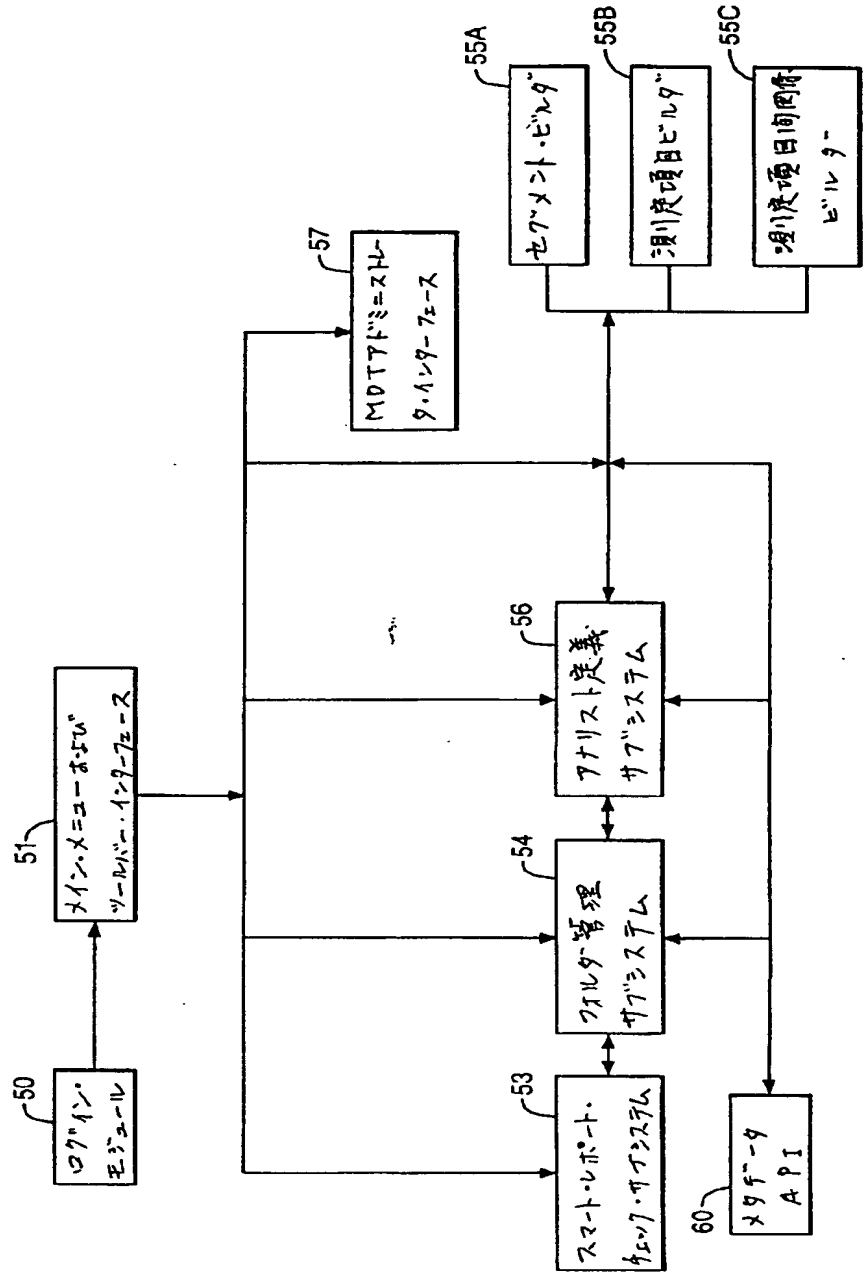
【図２１】



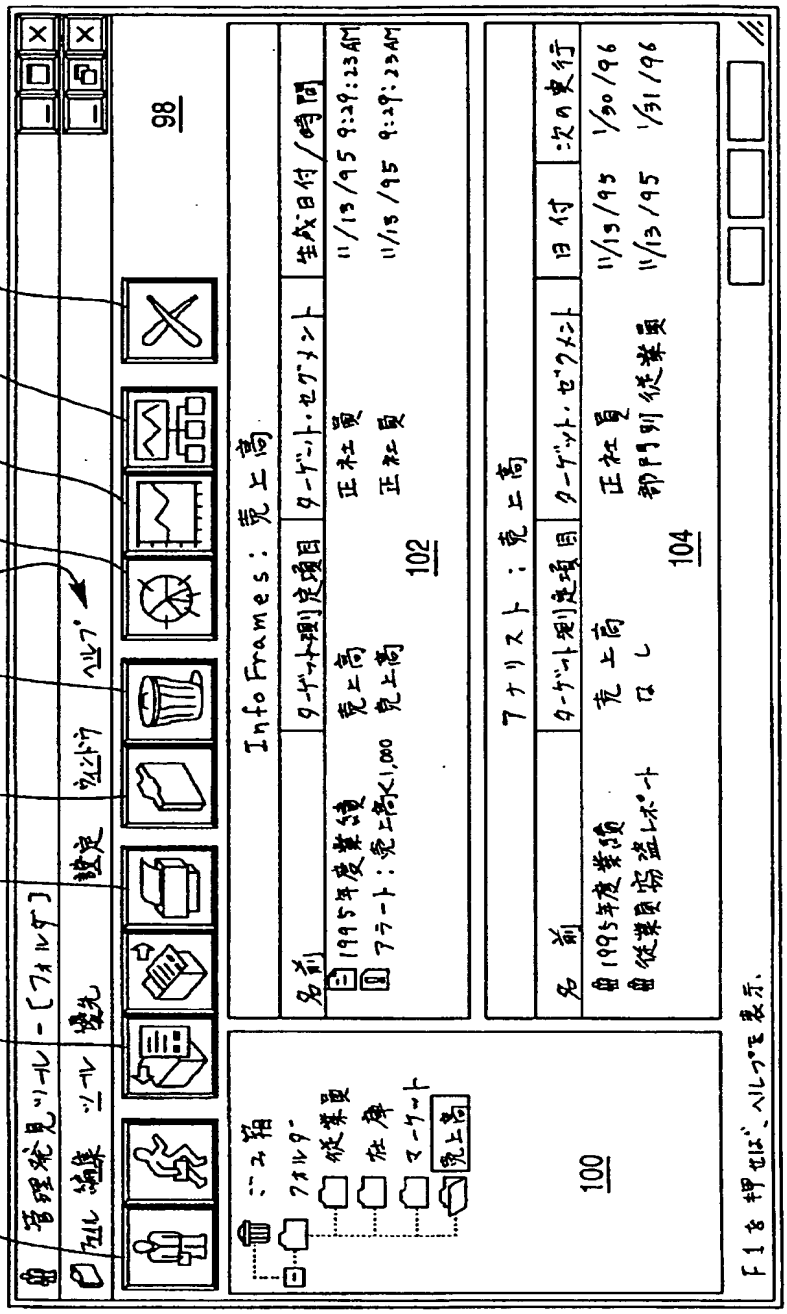
【図1】


データ・ウェアハウス
メタデータ
24
25
34
DSMサブ・システム
16
32
DAIサブ・システム
14
スケジューラ・サブシステム
18
クライアント・サブシステム
12
30
10
入力装置
21
ディスプレイ
22
プリンタ
23
記憶装置
26

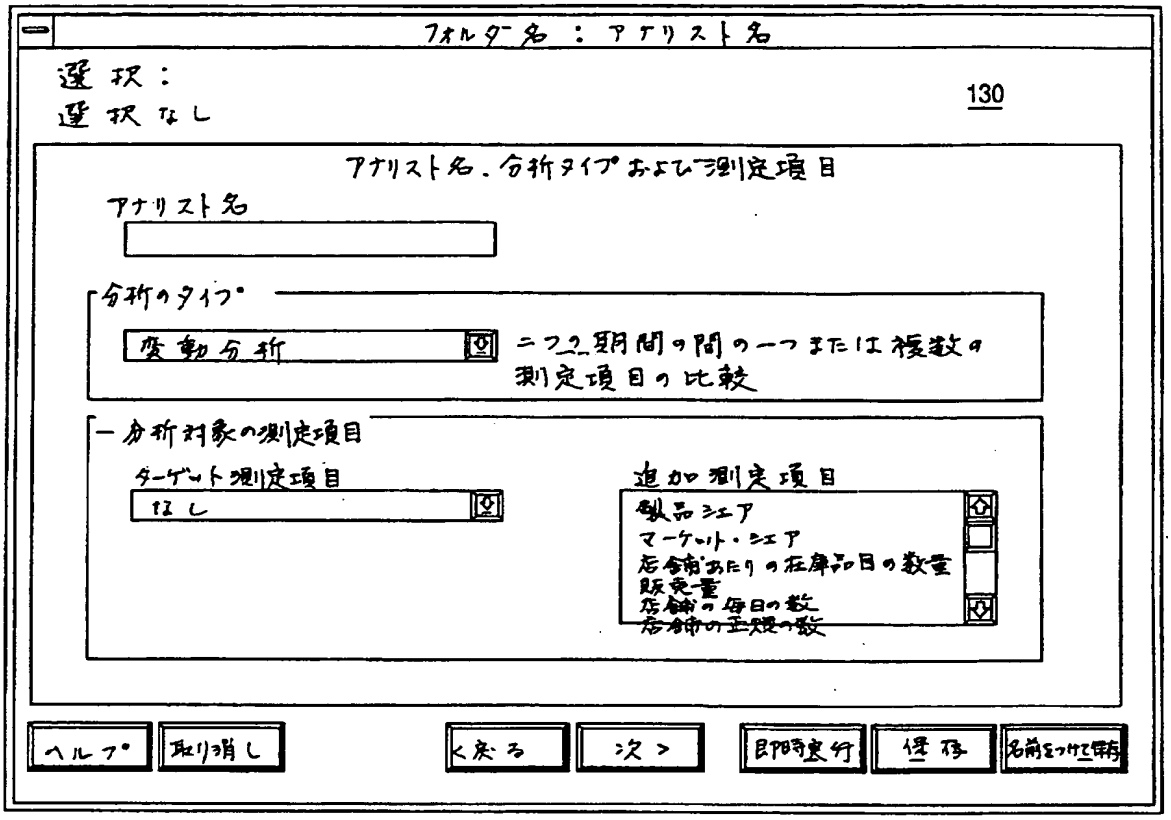
【図2】


55A
55B
55C
セグメント・ビルダ
項目ビルダ
項目間関係ビルダ
57
MDTアドミニストレータ・インターフェース
56
アナリスト定義サブシステム
54
ワイルダ管理サブシステム
53
スマート・レポート・チェック・サブシステム
51
メイン・メニュー・オブジェクト・インターフェース
50
ログイン・モジュール
60
メタデータAPI

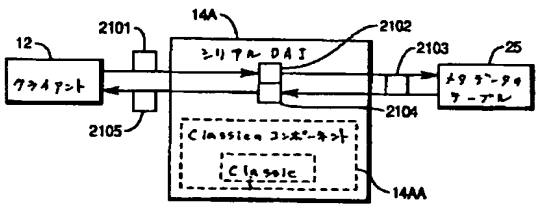
【図6】


管理発見ツール - [フォルダ]
ファイル 編集 ビュー 検索 設定 ウィンドウ ヘルプ
ごみ箱
フォルダ
従業員
在庫
マーケット
売上高
100
Info Frames: 売上高
名前
ターゲット規則定義項目
ターゲット・セグメント
生成日付/時間
1995年度業績
アラート：売上高<1,000
売上高
売上高
正社員
正社員
11/13/95 9:29:23AM
11/13/95 9:29:23AM
102
アナリスト：売上高
名前
ターゲット規則定義項目
ターゲット・セグメント
日付
次の実行
1995年度業績
従業員窃盗レポート
売上高
なし
正社員
都門別従業員
11/13/95
11/13/95
1/30/96
1/31/96
104
98
F1を押せば、ヘルプを表示。
110
120
150
118
116
106
112
157
152
122

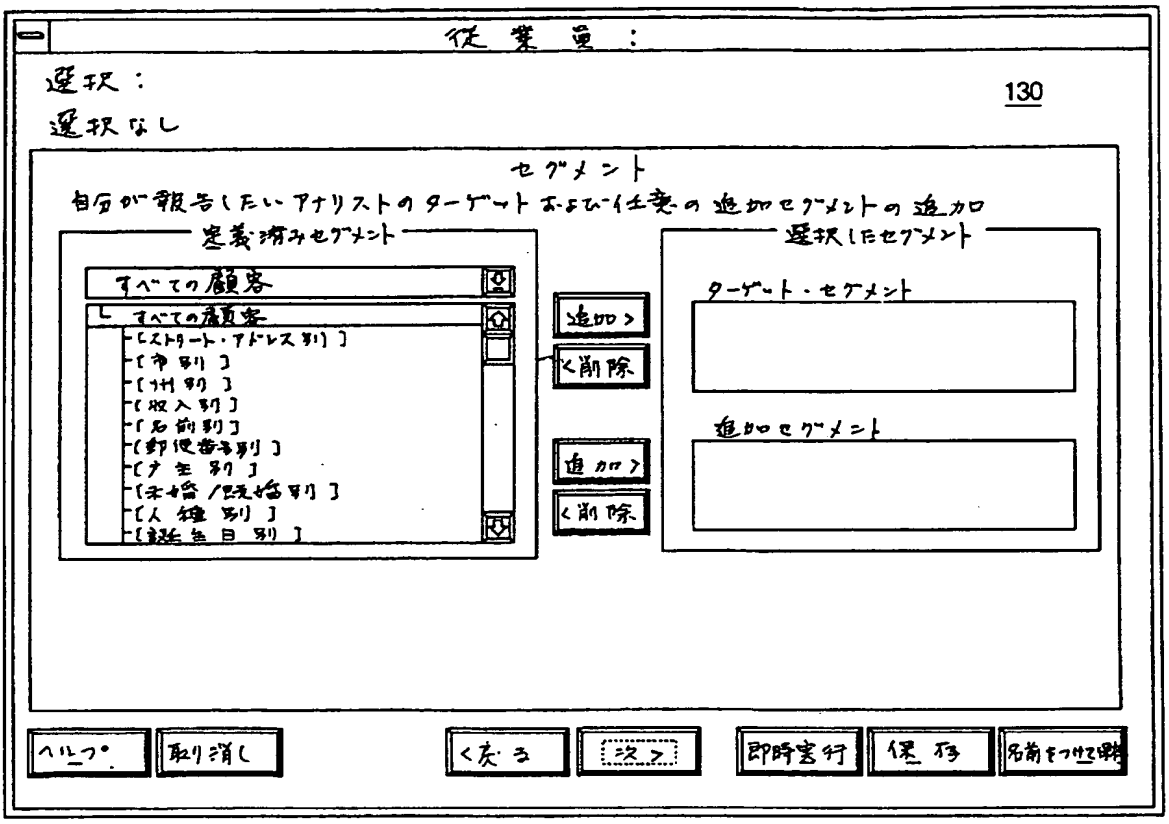
【図7】


フォルダ名 ： アナリスト名
選択：
選択なし
130
アナリスト名、分析タイプおよび測定項目
アナリスト名
分析のタイプ
変動分析
二つの期間の間の一つまたは複数の測定項目の比較
分析対象の測定項目
ターゲット測定項目
なし
追加測定項目
製品シェア
マーケット・シェア
店舗あたりの在庫品目の数量
販売量
店舗の毎日の数
店舗の正規の数
ヘルプ
取り消し
< 戻る
次 >
即時実行
保存
名前をつけて保存

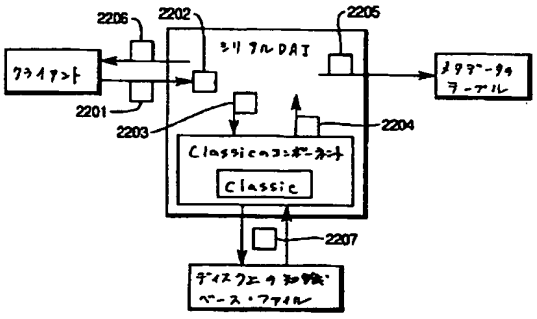
【図28】


12
クライアント
2101
2105
14A
シリアル DAI
Classic コンポーネント
Classic
14AA
2102
2103
2104
25
メタデータ ターゲル

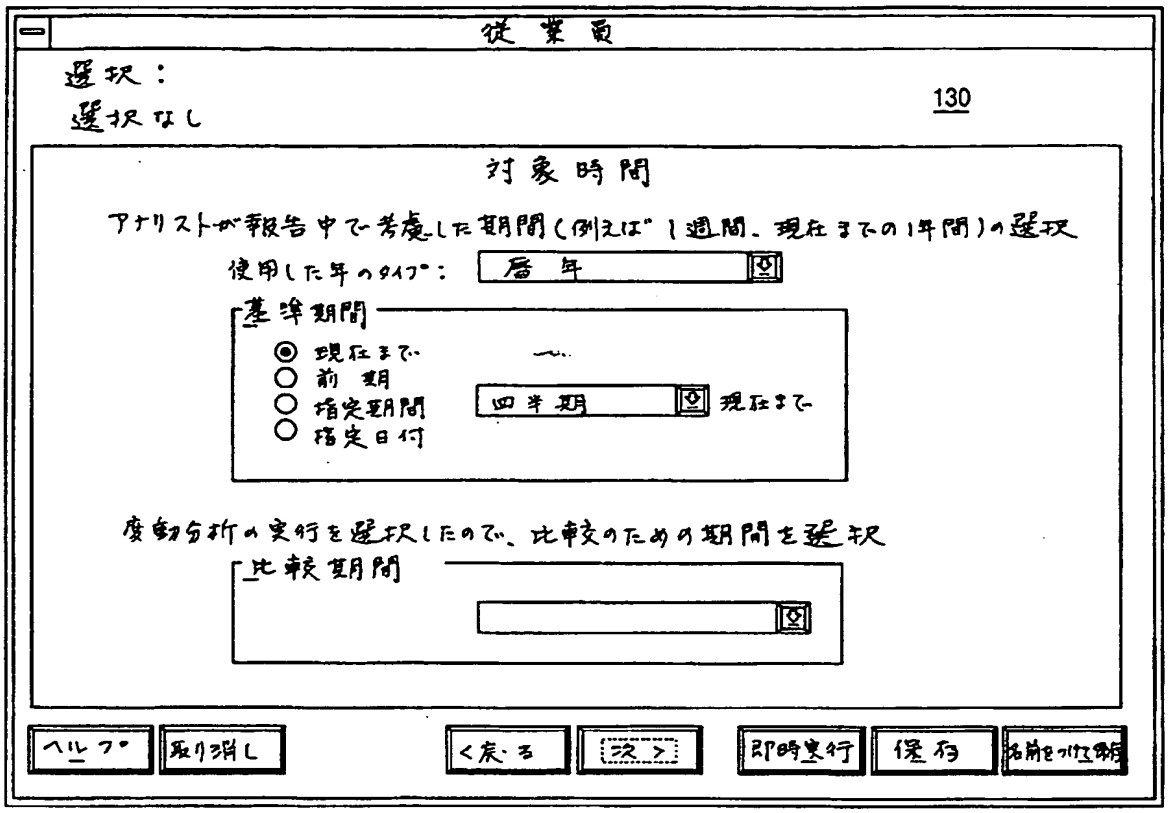
【図8】


従業員：
選択：
選択なし
130
セグメント
自分が報告したいアナリストのターゲットおよび任意の追加セグメントの追加
定義済みセグメント
すべての顧客
すべての顧客
[ストリート・アドレス別]
[市別]
[州別]
[収入別]
[名前別]
[郵便番号別]
[性別]
[未婚/既婚別]
[人種別]
[誕生日別]
追加>
<削除
追加>
<削除
選択したセグメント
ターゲット・セグメント
追加セグメント
ヘルプ
取り消し
<戻る
次>
即時実行
保存
名前をつけて保存

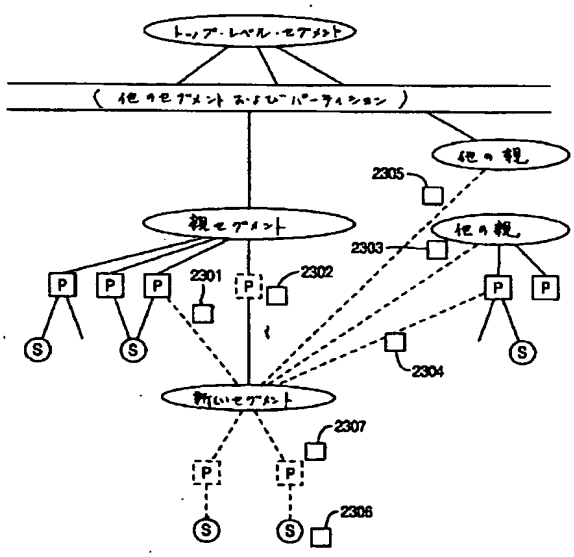
【図29】


2206
2202
2205
シリアルDAT
クライアント
メタデータ
ファイル
2201
2203
2204
Classicコンポーネント
Classic
2207
ディスクエクステンド
ベース・ファイル

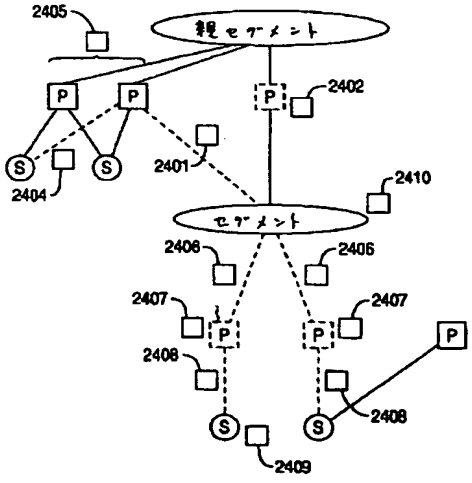
【図9】


従業員
選択：
選択なし
130
対象時間
アナリストが報告中で考慮した期間（例えば"1週間、現在までの1年間）の選択
使用した年のタイプ：
暦年
基準期間
現在まで
前期
指定期間
指定日付
四半期
現在まで
変動分析の実行を選択したので、比較のための期間を選択
比較期間
ヘルプ
取り消し
<戻る
次>
即時実行
保存
名前をつけて保存

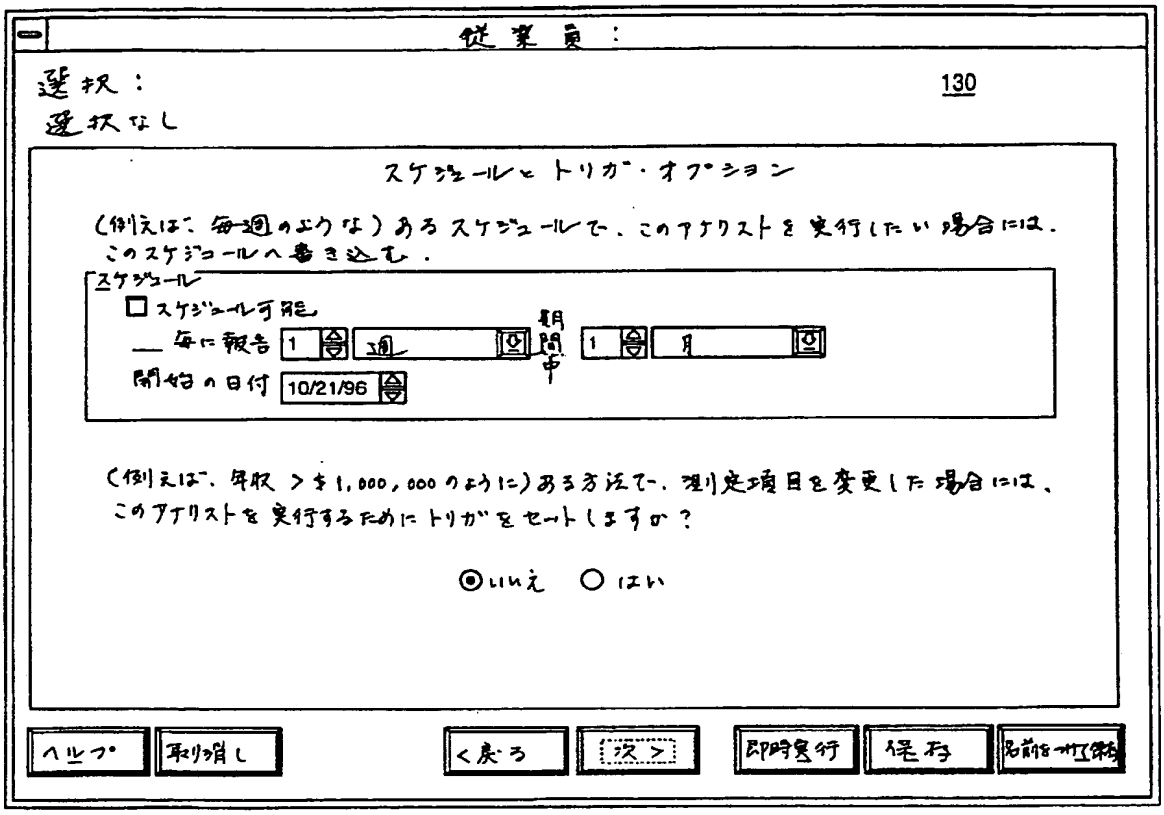
【図30】


トップ・レベル・セグメント
（他のセグメントおよびパーティション）
他の親
2305
親セグメント
他の親
2303
2301
2302
2304
新しいセグメント
2307
2306

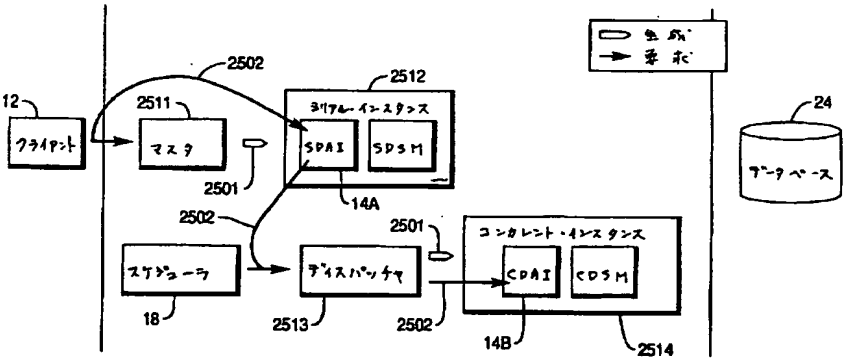
【図31】


2405
親セグメント
2402
2401
2404
2410
セグメント
2406
2406
2407
2407
2408
2408
2409

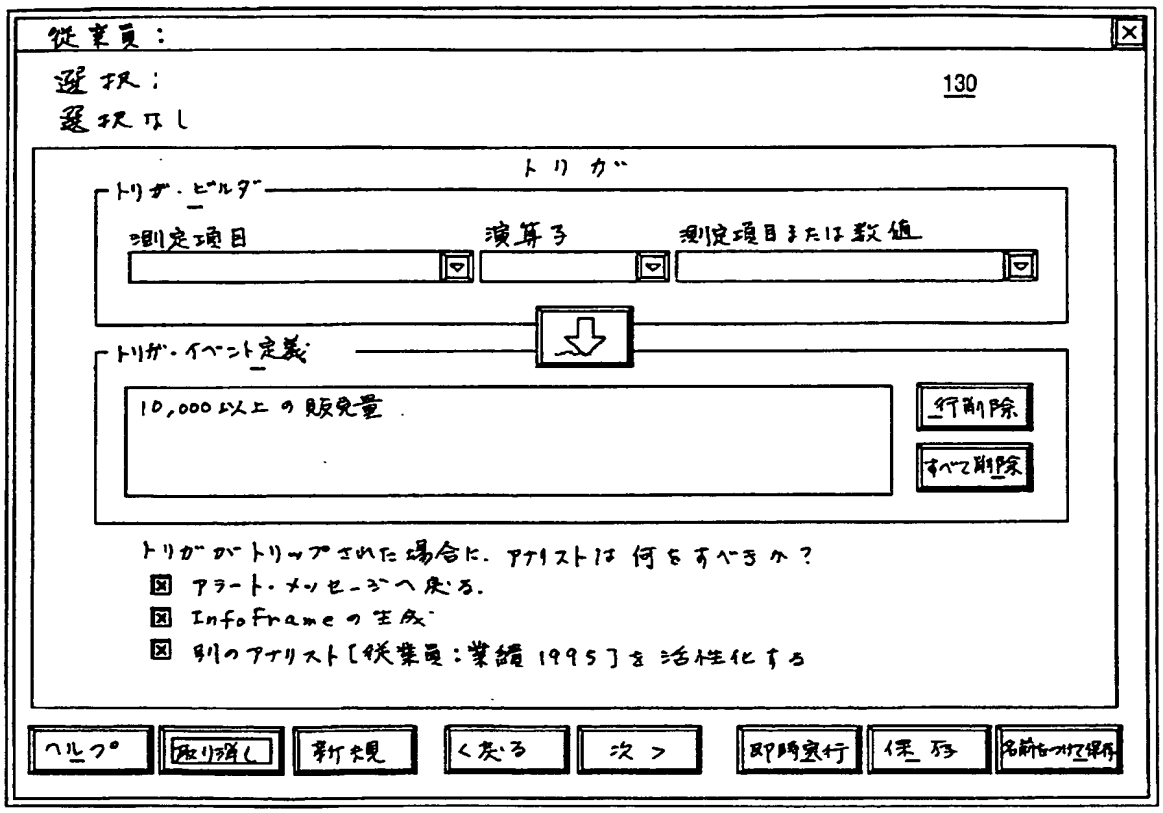
【図10】


従業員：
選択：
130
選択なし
スケジュールとトリガ・オプション
（例えば、毎週のような）あるスケジュールで、このアナリストを実行したい場合には、このスケジュールへ書き込む。
スケジュール
スケジュール可能
毎に報告
1
週
期間中
1
月
開始の日付
10/21/96
（例えば、年収 > $1,000,000のように）ある方法で、測定項目を変更した場合には、このアナリストを実行するためにトリガをセットしますか？
いいえ
はい
ヘルプ
取り消し
< 戻る
次 >
即時実行
保存
名前を付けて保存

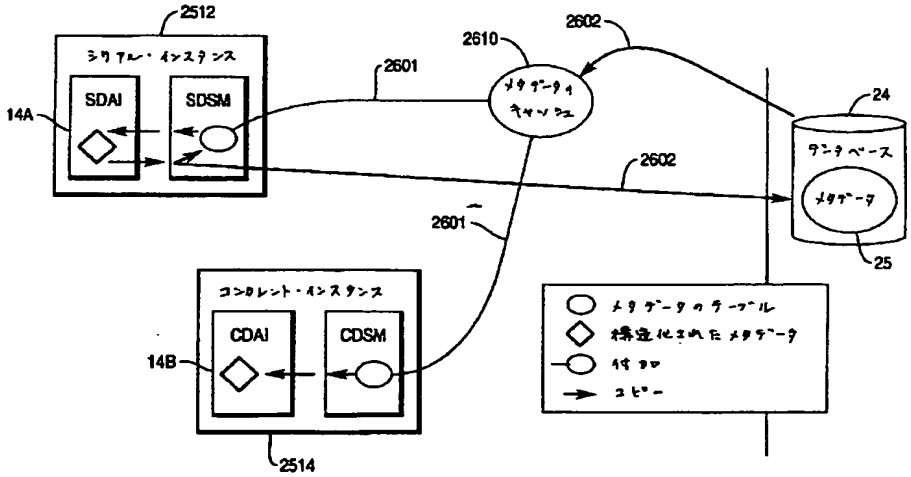
【図32】


生成
参照
12
クライアント
2502
2511
マスタ
2501
2512
SPAI
SPSM
14A
24
データベース
2502
スケジューラ
18
ディスパッチャ
2513
2501
2502
コンカレント・インスタンス
CDAI
CDSM
14B
2514

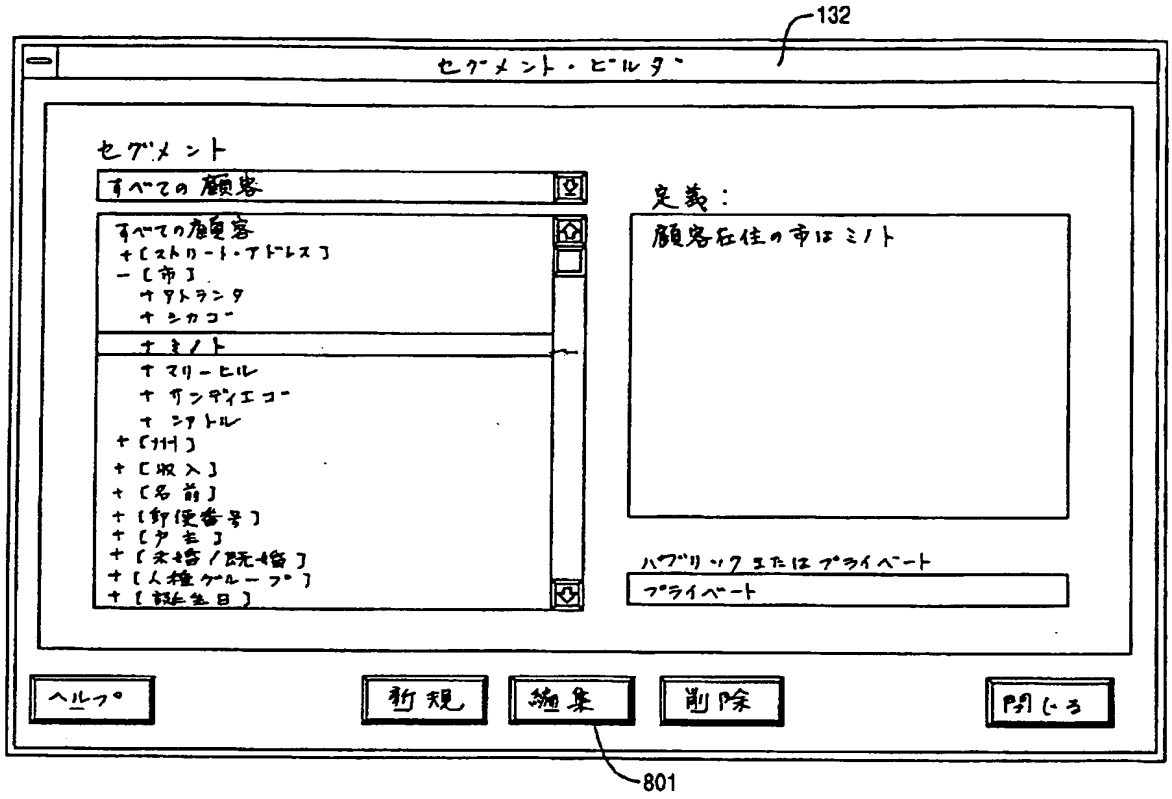
【図11】


従業員：
選択：
選択なし
130
トリガ
トリガ・ビルダー
測定項目
演算子
測定項目または数値
トリガ・イベント定義
10,000以上の販売量
行削除
すべて削除
トリガがトリップされた場合に、アナリストは何をすべきか？
アラート・メッセージを表示する。
InfoFrameの生成
別のアナリスト［従業員：業績 1995］を活性化する
ヘルプ
取り消し
新規
＜戻る
次＞
即時実行
保存
名前をつけて保存

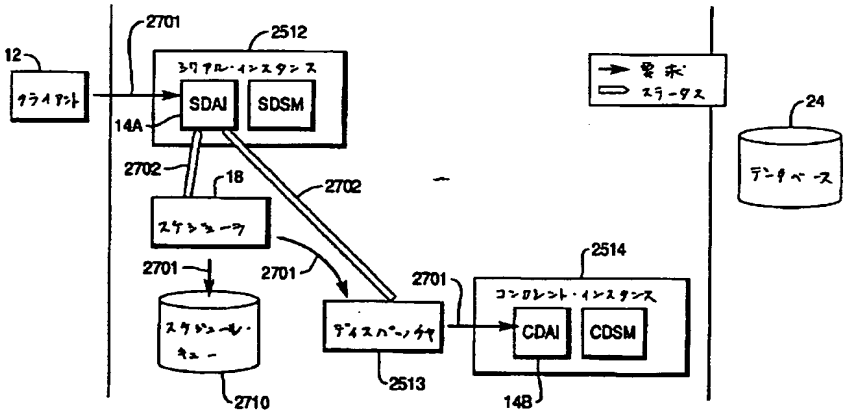
【図33】


2512
シリアル・インスタンス
SDAI
SDSM
14A
2601
2610
メタデータ
キャッシュ
2602
24
データベース
メタデータ
25
2602
2601
コンカレント・インスタンス
CDAI
CDSM
14B
2514
メタデータのラベル
構造化されたメタデータ
付与
コピー

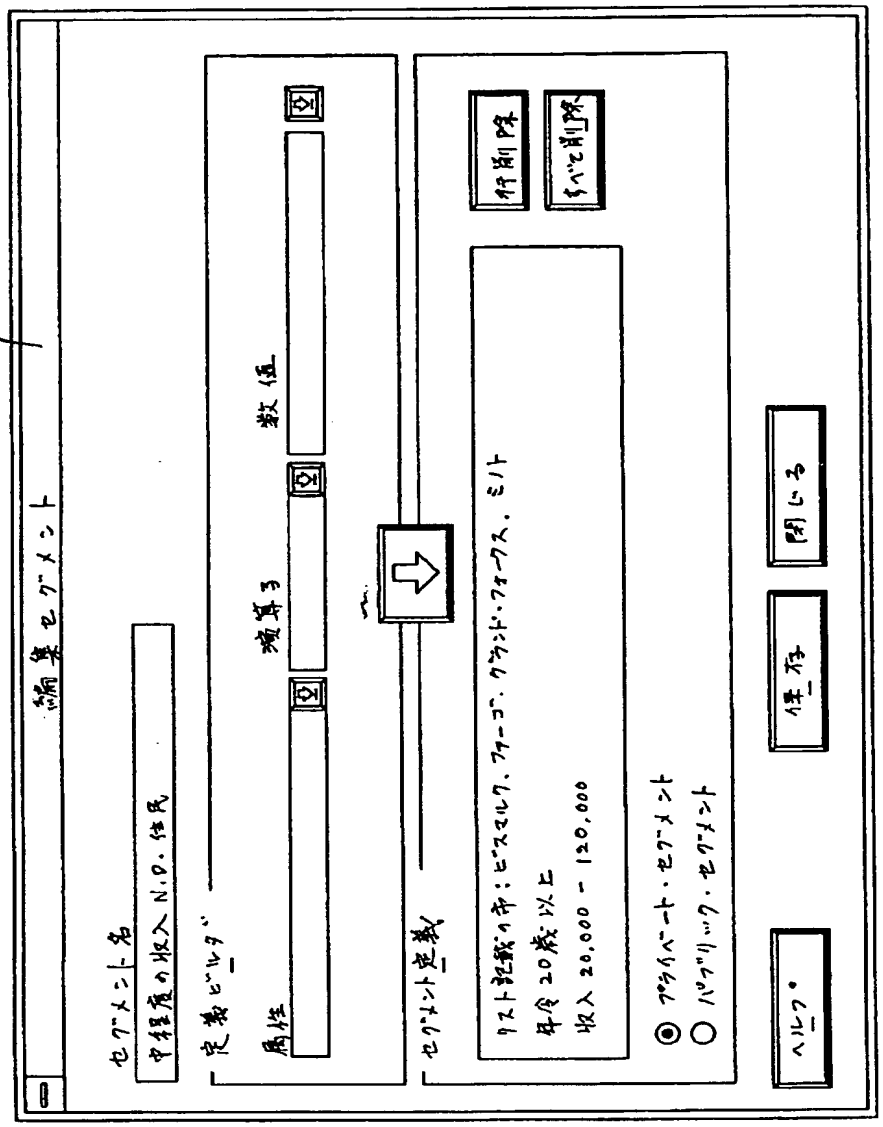
【図12】


132
セグメント・ビルダー
セグメント
すべての顧客
すべての顧客
＋［ストリート・アドレス］
－［市］
＋アトランタ
＋シカゴ
＋ミノト
＋マリーヒル
＋サンディエゴ
＋シアトル
＋［州］
＋［収入］
＋［名前］
＋［郵便番号］
＋［年齢］
＋［未婚／既婚］
＋［人種グループ］
＋［誕生日］
定義：
顧客在住の市はミノト
ハブリックまたはプライベート
プライベート
ヘルプ
新規
編集
削除
閉じる
801

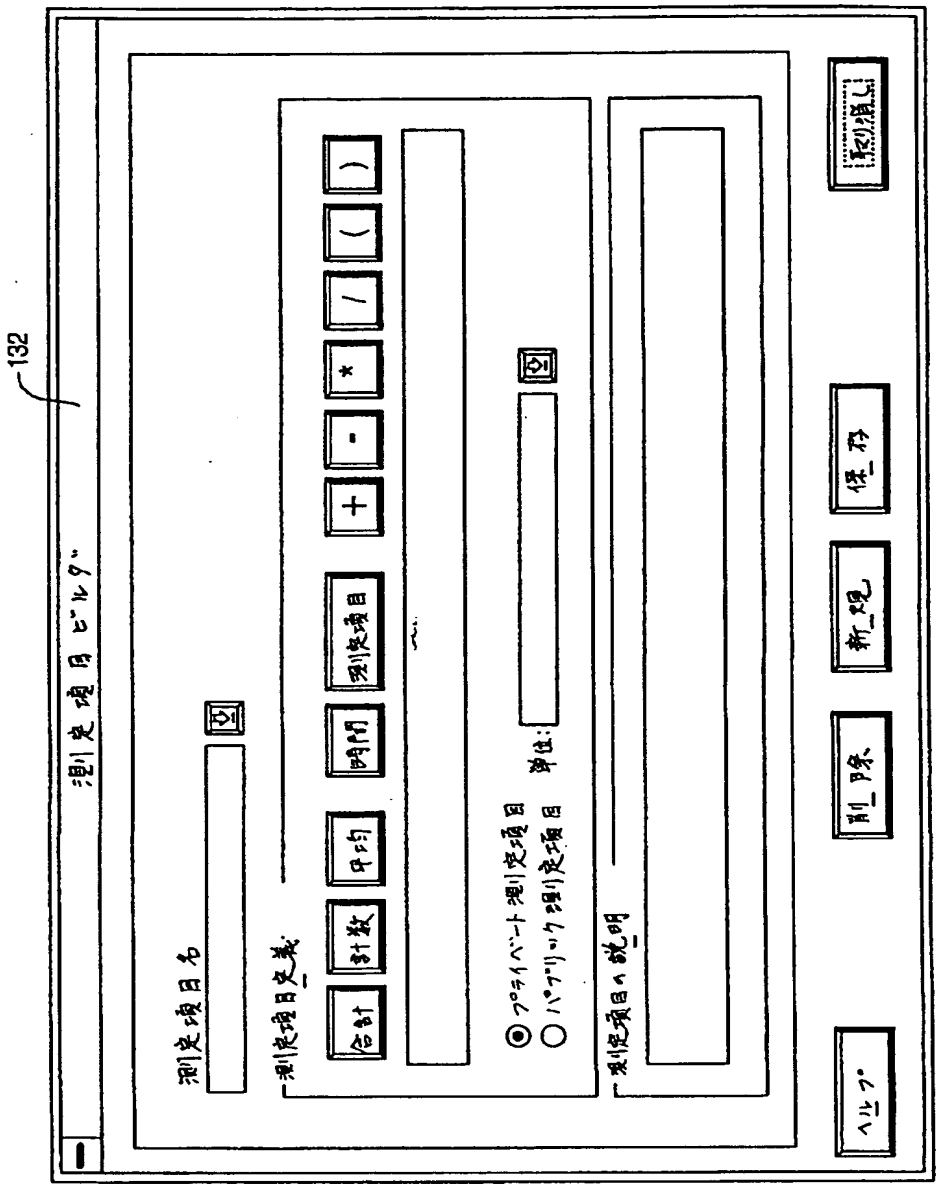
【図34】


12
クライアント
2701
2512
コアル・インスタンス
SDAI
SDSM
14A
2702
18
スケジューラ
2702
2701
2701
スケジュール・キュー
2710
ディスパッチャ
2513
2701
2514
コンカレント・インスタンス
CDAI
CDSM
14B
要求
エラータス
24
データベース

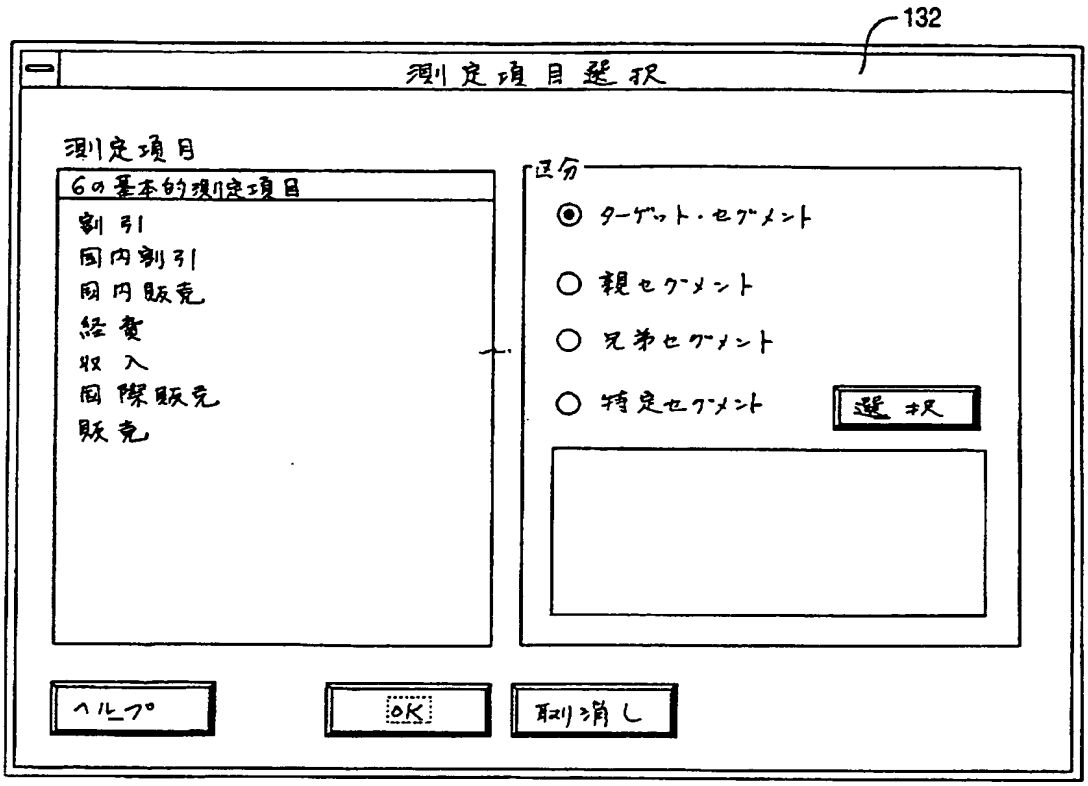
【図13】


132
編集セグメント
セグメント名
中程度の収入 N.O.住民
定義ビルダー
属性
演算子
数値
セグメント定義
リスト記載の市：ビスマルク、フォーゴ、グランド・フォークス、ミノト
年令 20歳以上
収入 20,000 - 120,000
◉ アクティベート・セグメント
○ バックリンク・セグメント
行削除
すべて削除
ヘルプ
保存
閉じる

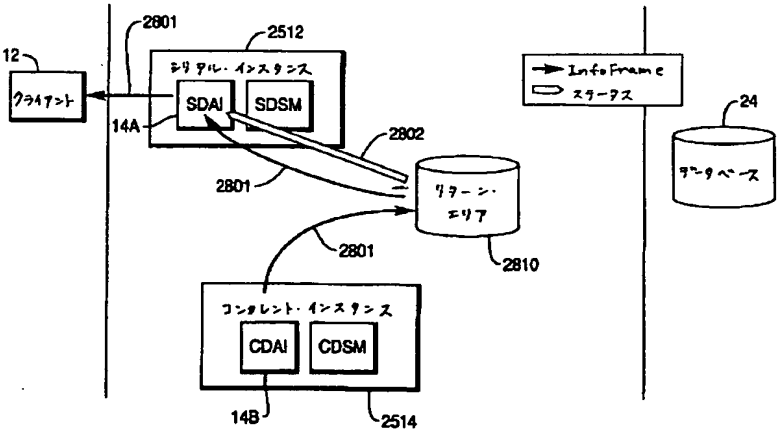
【図14】


132
測定項目ビルダ
測定項目名
測定項目定義
合計
計数
平均
時間
測定項目
+
-
*
/
(
)
プライベート測定項目
パブリック測定項目
単位:
測定項目の説明
ヘルプ
削除
新規
保存
測定項目

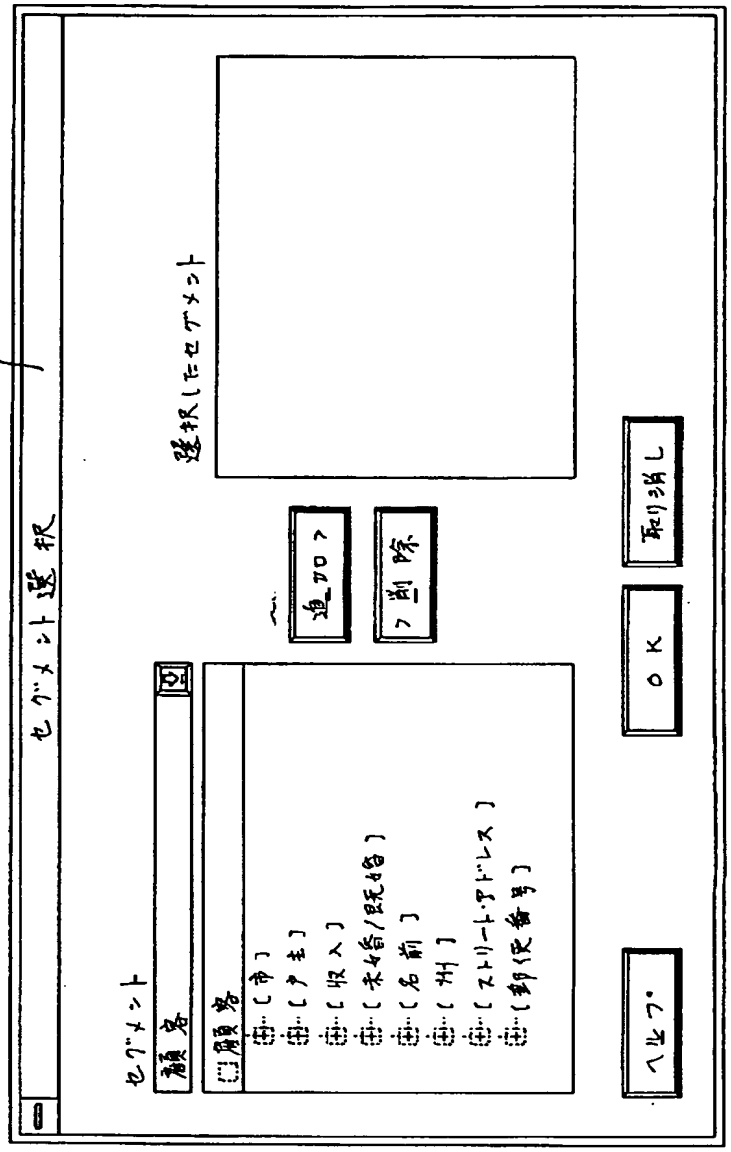
【図15】


132
測定項目選択
測定項目
6の基本的測定項目
割引
国内割引
国内販売
経費
収入
国際販売
販売
区分
ターゲット・セグメント
親セグメント
兄弟セグメント
特定セグメント
選択
ヘルプ
OK
取り消し

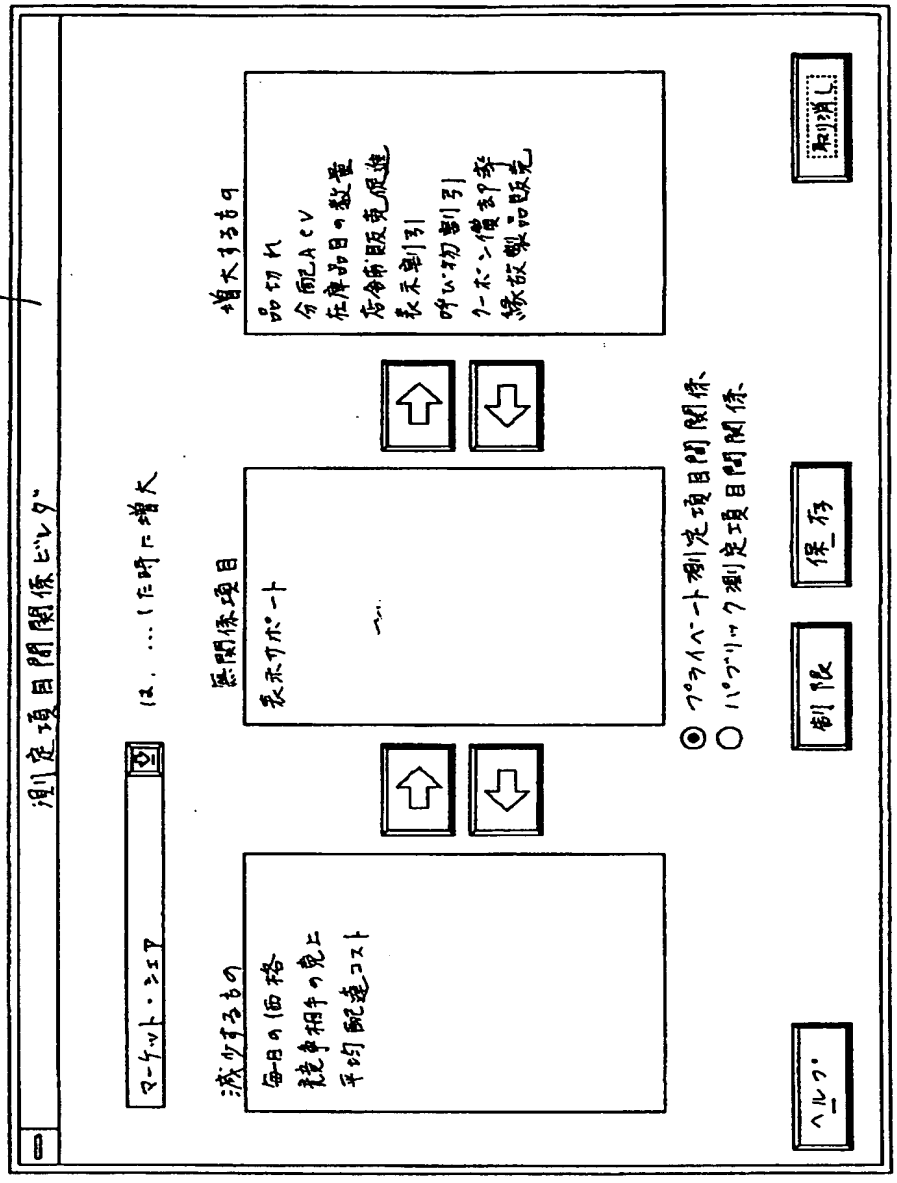
【図35】


12
クライアント
2801
2512
シリアル・インスタンス
SDAI
SDSM
14A
2802
2801
リターン・エリア
2810
2801
コンテント・インスタンス
CDAI
CDSM
14B
2514
InfoFrame
スケース
24
データベース

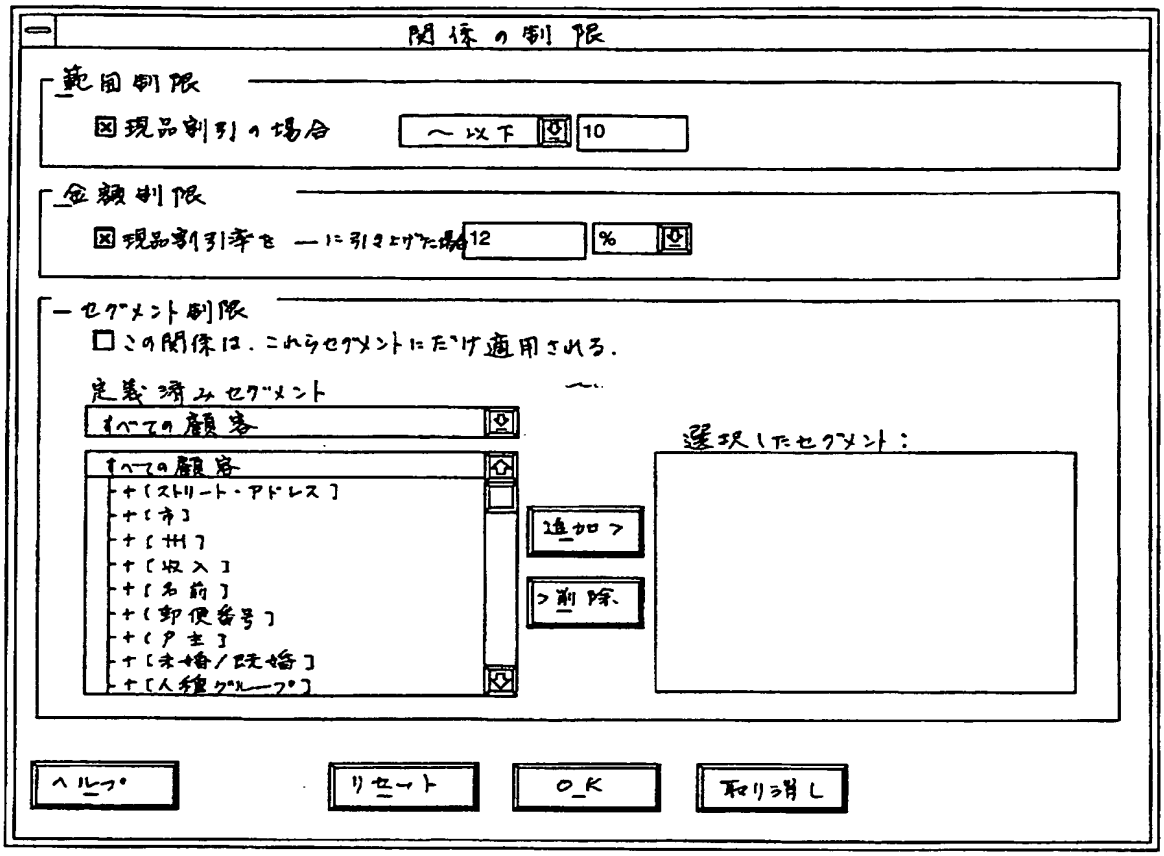
【図16】


132
セグメント選択
セグメント
顧客
〔市〕
〔ノキ〕
〔収入〕
〔未婚/既婚〕
〔名前〕
〔姓〕
〔ストリート・アドレス〕
〔郵便番号〕
追加>>
<削除
選択したセグメント
ヘルプ
OK
取り消し

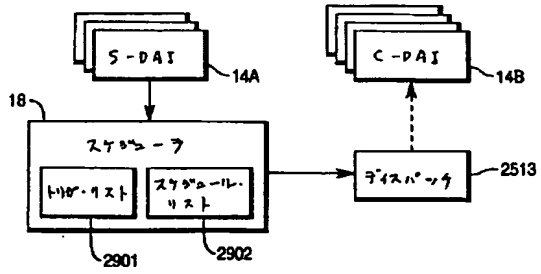
【図17】


132
測定項目関係ビルダー
マーケット・シェア
は、…に比例に増大
減少するもの
毎日の価格
競争相手の売上
平均配達コスト
無関係項目
表ホワポート
増大するもの
品切れ
分配レベル
在庫品目の数量
危険販売促進
表示割引
吟い物割引
クーポン備考印等
係数製品販売
グラフベート測定項目関係
バーグリック測定項目関係
ヘルプ
削除
保存
取消し

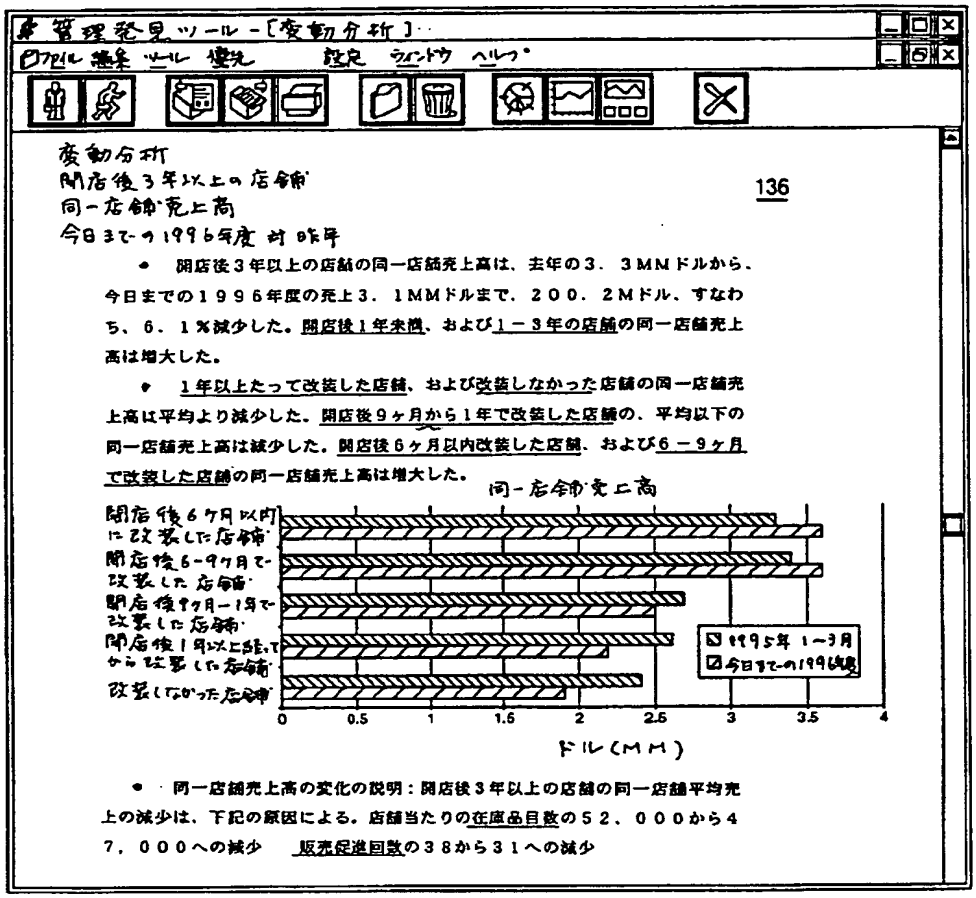
【図18】


関係の制限
範囲制限
現品割引の場合
～以下
10
金額制限
現品割引率を
に引き上げた場合
12
%
セグメント制限
この関係は、これらセグメントにだけ適用される。
定義済みセグメント
すべての顧客
すべての顧客
＋［ストリート・アドレス］
＋［市］
＋［州］
＋［収入］
＋［名前］
＋［郵便番号］
＋［戸主］
＋［未婚／既婚］
＋［人種グループ］
追加＞
＞削除
選択したセグメント：
ヘルプ
リセット
OK
取り消し

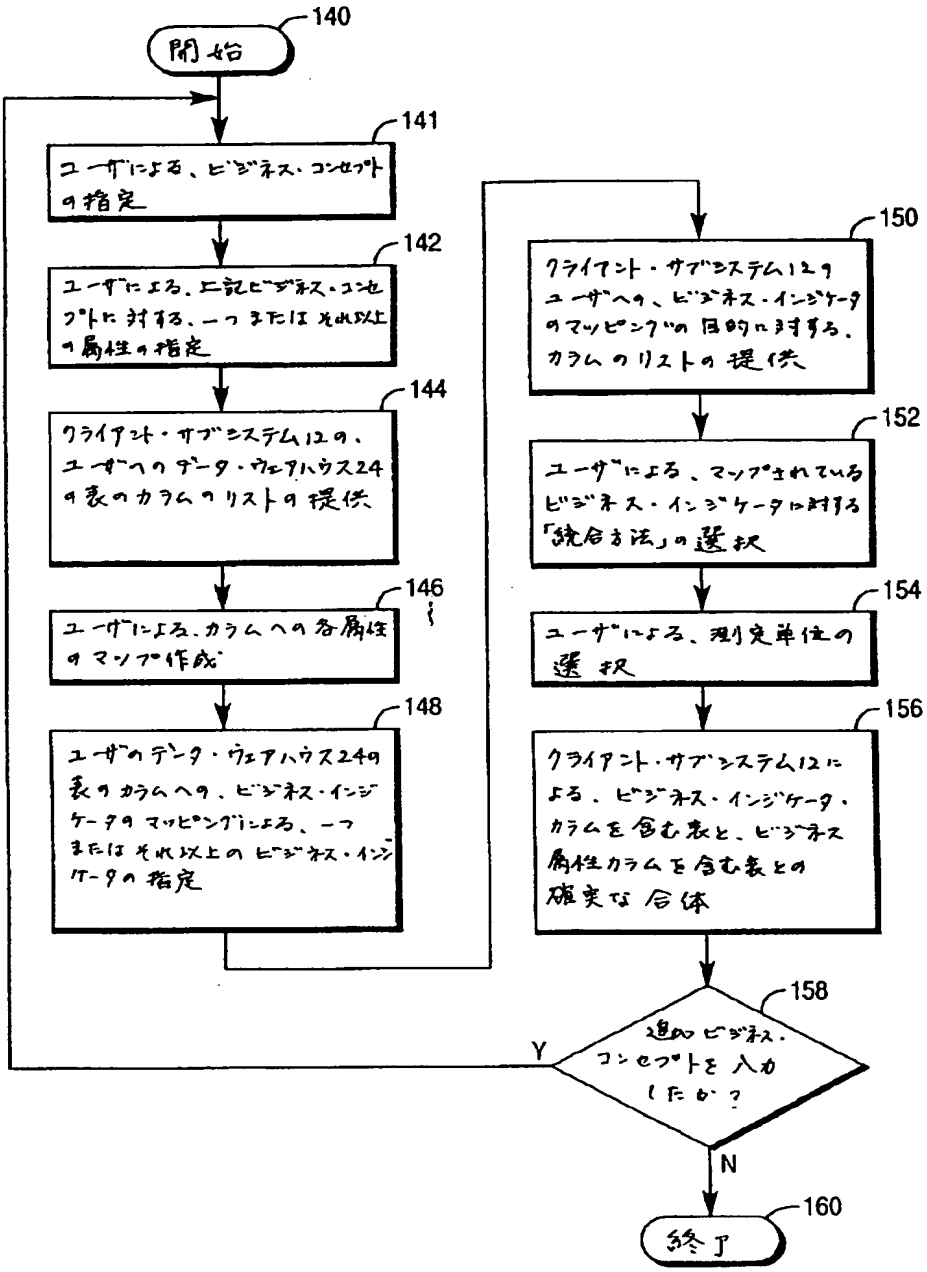
【図36】


S-DAI
14A
C-DAI
14B
18
スケジューラ
トリガ・リスト
スケジュール・リスト
ディスパッチ
2513
2901
2902

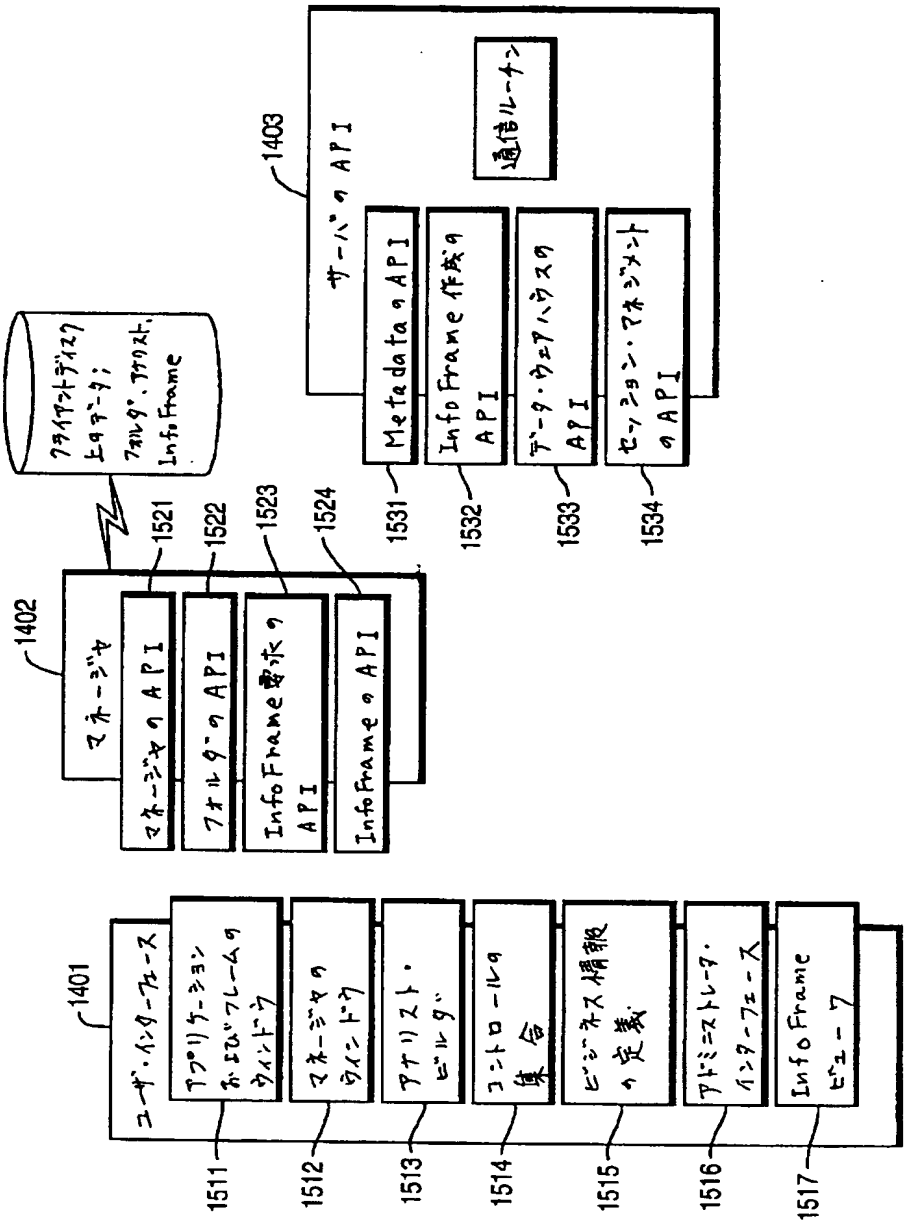
【図19】


管理発見ツール－[変動分析]
ファイル 編集 ツール 優先 設定 ウィンドウ ヘルプ
変動分析
開店後3年以上の店舗
同一店舗売上高
今日までの1996年度 対 昨年
136
• 開店後3年以上の店舗の同一店舗売上高は、去年の3．3MMドルから、今日までの1996年度の売上3．1MMドルまで、200．2Mドル、すなわち、6．1%減少した。開店後1年未満、および1－3年の店舗の同一店舗売上高は増大した。
• 1年以上たって改装した店舗、および改装しなかった店舗の同一店舗売上高は平均より減少した。開店後9ヶ月から1年で改装した店舗の、平均以下の同一店舗売上高は減少した。開店後6ヶ月以内改装した店舗、および6－9ヶ月で改装した店舗の同一店舗売上高は増大した。
同一店舗売上高
開店後6ヶ月以内に改装した店舗
開店後6-9ヶ月で改装した店舗
開店後9ヶ月-1年で改装した店舗
開店後1年以上たってから改装した店舗
改装しなかった店舗
1995年 1-3月
今日までの1996年度
0
0.5
1
1.5
2
2.5
3
3.5
4
ドル(MM)
• 同一店舗売上高の変化の説明：開店後3年以上の店舗の同一店舗平均売上の減少は、下記の原因による。店舗当たりの在庫品目数の52，000から47，000への減少　販売促進回数の38から31への減少

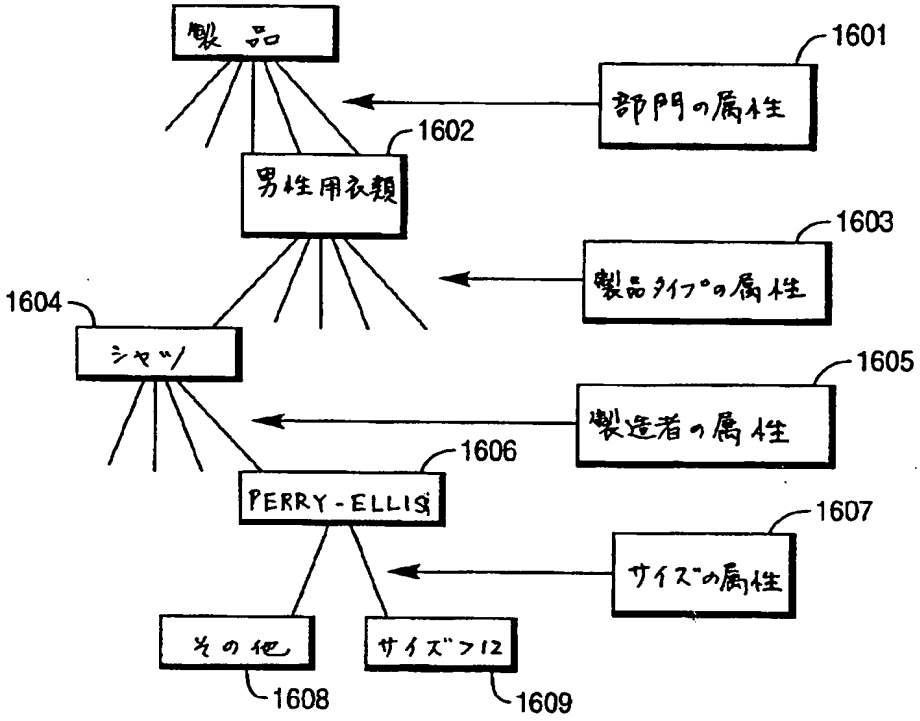
【図20】


140
開始
141
ユーザによる、ビジネス・コンセプトの指定
142
ユーザによる、上記ビジネス・コンセプトに対する、一つまたはそれ以上の属性の指定
144
クライアント・サブシステム12の、ユーザへのデータ・ウェアハウス24の表のカラムのリストの提供
146
ユーザによる、カラムへの各属性のマップ作成
148
ユーザのデータ・ウェアハウス24の表のカラムへの、ビジネス・インジケータのマッピングによる、一つまたはそれ以上のビジネス・インジケータの指定
150
クライアント・サブシステム12のユーザへの、ビジネス・インジケータのマッピングの目的にマッチする、カラムのリストの提供
152
ユーザによる、マップされているビジネス・インジケータに対する「統合方法」の選択
154
ユーザによる、測定単位の選択
156
クライアント・サブシステム12による、ビジネス・インジケータ、カラムを含む表と、ビジネス属性カラムを含む表との確実な合体
158
追加ビジネス・コンセプトを入力したか？
Y
N
160
終了

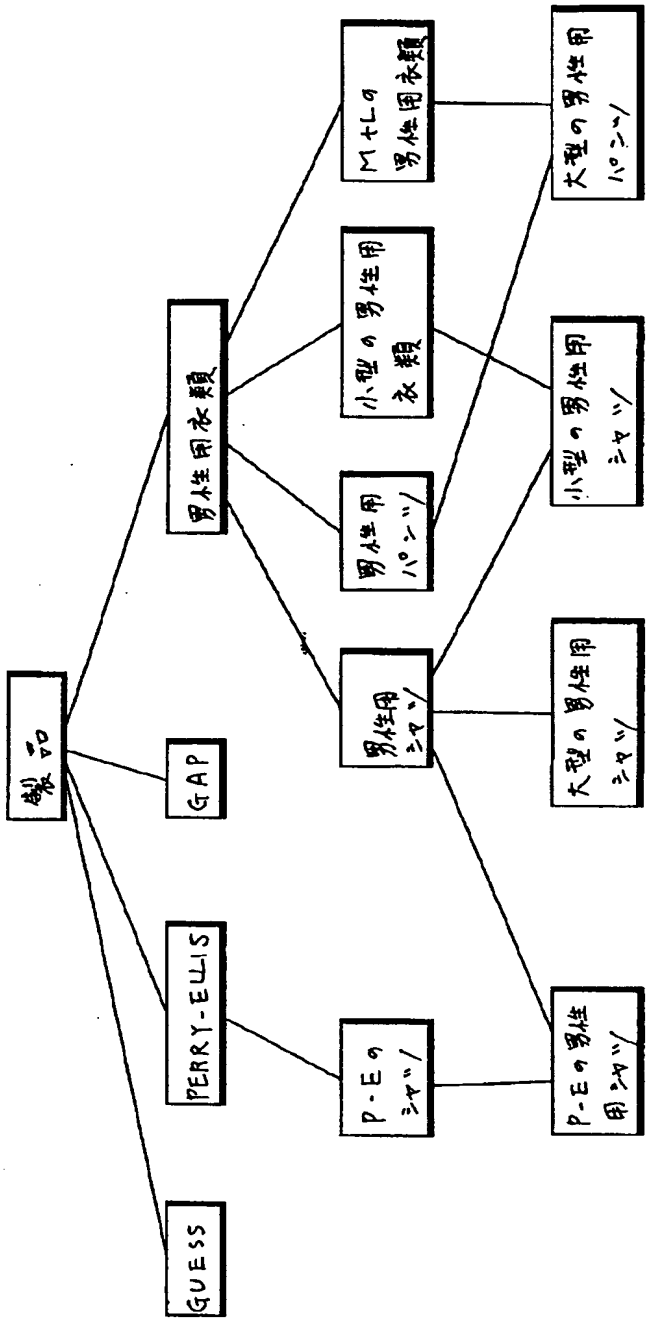
【図22】


1401
ユーザ・インターフェース
アプリケーション および フレームの ウィンドウ
マネージャの ウィンドウ
アナリスト・ ビルダ
コントロールの 集合
ビジネス情報 の定義
アドミニストレータ・ インターフェース
InfoFrame ビュー
1511
1512
1513
1514
1515
1516
1517
1402
マネージャ
マネージャの API
フォルダの API
InfoFrame要求の API
InfoFrameの API
1521
1522
1523
1524
クライアントディスク 上のデータ; フォルダ, アウトライン, InfoFrame
1403
サーバの API
Metadata の API
InfoFrame 作成の API
データ・ウェアハウスの API
セッション・マネジメントの API
通信ルーチン
1531
1532
1533
1534

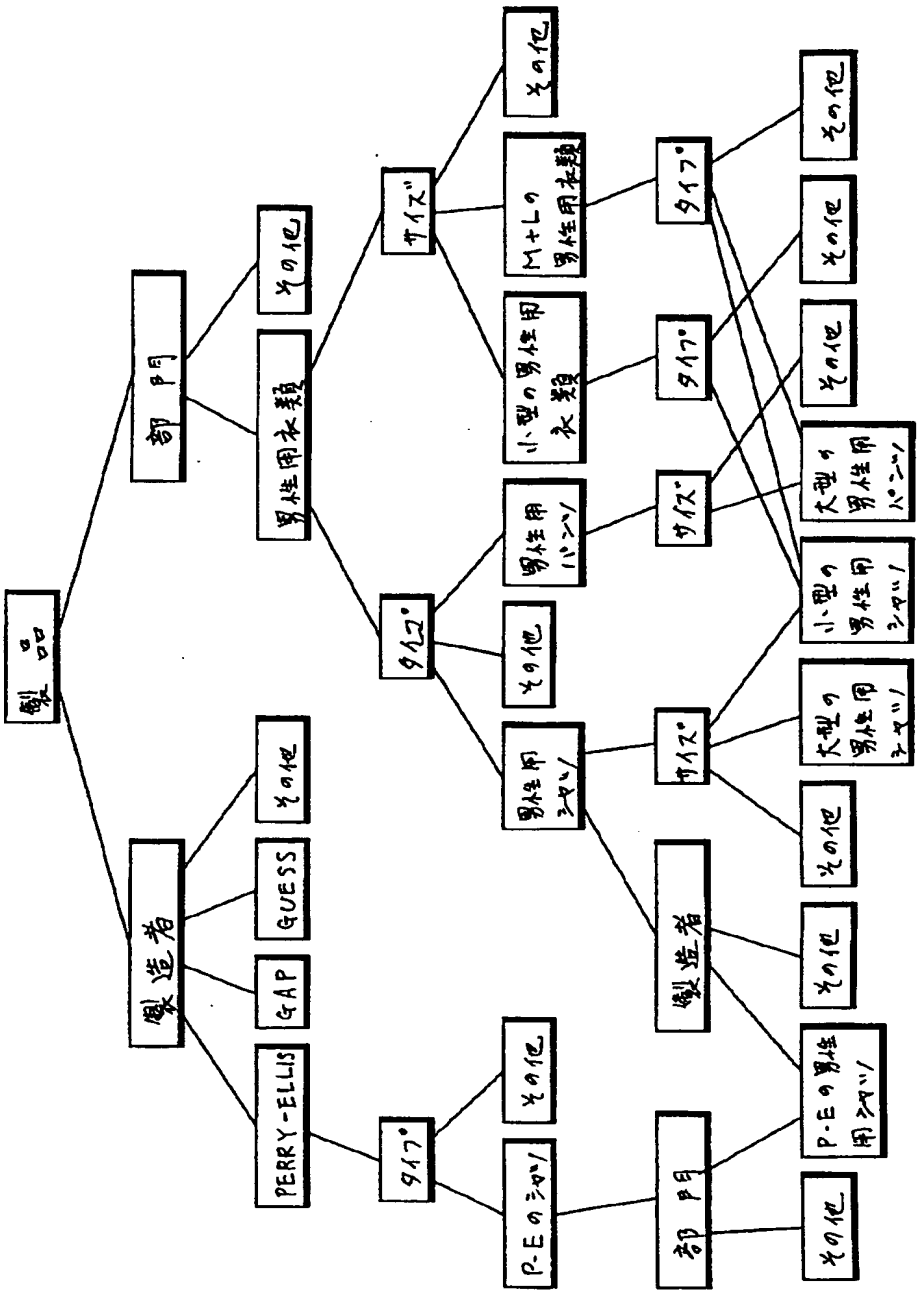
【図23】


製品
1601
部門の属性
1602
男性用衣類
1603
製品タイプの属性
1604
シャツ
1605
製造者の属性
1606
PERRY-ELLIS
1607
サイズの属性
その他
サイズ 12
1608
1609

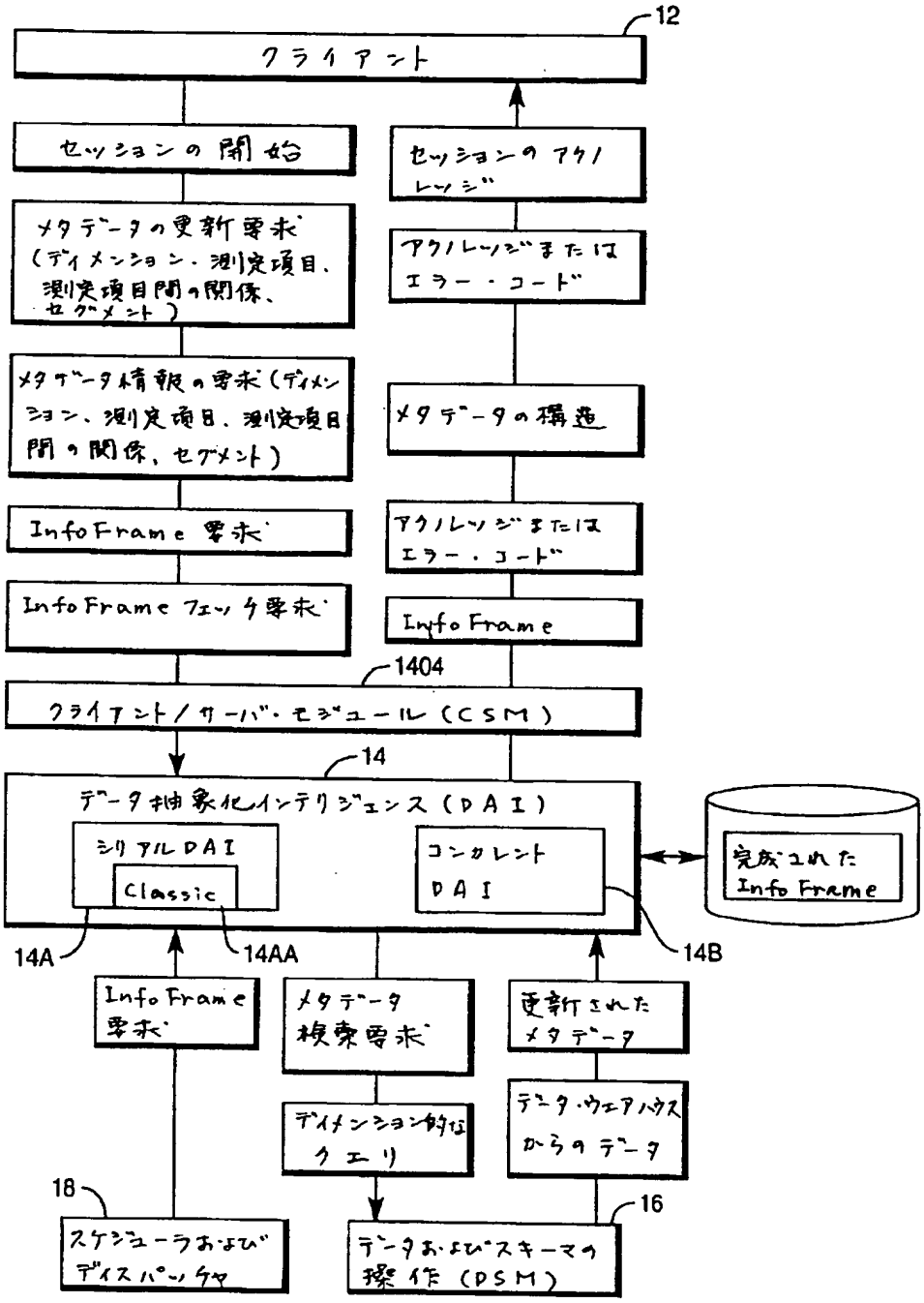
【図24】


製品
GAP
PERRY-ELLIS
GUESS
男性用衣類
男性用シャツ
P-Eのシャツ
M+Lの男性用衣類
小型の男性用衣類
男性用パンツ
大型の男性用パンツ
小型の男性用シャツ
大型の男性用シャツ
P-Eの男性用シャツ

【図25】


製品
部門
製造者
男性用衣類
その他
GAP
GUESS
PERRY-ELLIS
サイズ
タイプ
その他
M+Lの男性用衣類
小型の男性用衣類
男性用パンツ
その他
男性用シャツ
P-Eのシャツ
その他
タイプ
サイズ
その他
大型の男性用パンツ
小型の男性用シャツ
大型の男性用シャツ
その他
P-Eの男性用シャツ
その他
製造者
部門
その他

【図26】


12
クライアント
セッションの開始
セッションのアクノレッジ
メタデータの更新要求（ディメンション、測定項目、測定項目間の関係、セグメント）
アクノレッジまたはエラー・コード
メタデータ情報の要求（ディメンション、測定項目、測定項目間の関係、セグメント）
メタデータの構造
InfoFrame 要求
アクノレッジまたはエラー・コード
InfoFrame フェッチ要求
InfoFrame
1404
クライアント／サーバ・モジュール（CSM）
14
データ抽象化インテリジェンス（DAI）
シリアルDAI
Classic
コンカレントDAI
完成されたInfo Frame
14A
14AA
14B
InfoFrame要求
メタデータ検索要求
更新されたメタデータ
ディメンション的なクエリ
データ・ウェアハウスからのデータ
18
スケジューラおよびディスパッチャ
16
データおよびスキーマの操作（DSM）

フロントページの続き

(72)発明者　グレン　キース　ウィクル
アメリカ合衆国　ニューメキシコ州　87505　サンタフェ　ボニート　ロード　3
(72)発明者　マーシャル　ポール　リンドセイ
アメリカ合衆国　カリフォルニア州　92131　サンディエゴ　エルマ　ロード、ナンバー　204、9949
(72)発明者　リチャード　ニール　シューベルト
アメリカ合衆国　カリフォルニア州　92131－1711　サンディエゴ　カミニト　バンヨン　10542
(72)発明者　ドゥルー　トーマス　レティントン
アメリカ合衆国　カリフォルニア州　92116－3939サンディエゴ　ノース　アヴェニュー　4494
(72)発明者　ジェフレイ　ポール　ラドウィグ
アメリカ合衆国　カリフォルニア州　92127　サンディエゴ　サニーデイル　コート　11250
(72)発明者　ジェイムズ　フォスター　クナッソン
アメリカ合衆国　サウスダコタ州　57049　ダコタ　ダンズ　コートヤード　ドライブ　230、アパートメント　207
(72)発明者　ソヘイラ　トヘリ
アメリカ合衆国　ジョージア州　30033　デカターエヌ．クロッシング　ウェイ　1610
(72)発明者　スコット　デイル　コウルター
アメリカ合衆国　ジョージア州　30060　マリエッタアイボリー　トレイル　3193
(72)発明者　ケヴィン　ワイン　コパス
アメリカ合衆国　ジョージア州　30045　ローレンスヴィレ　ロックランド　ウェイ　326

【外国語明細書】

1. Title of Invention

DATA MANAGEMENT SYSTEM

2. Claims

1. A database management system, characterized by:

(1) a database comprising data entities having associated attributes, wherein the data entities are arranged in a first heirarchy, each level of the first heirarchy being associated with at least one attribute;

(2) a server computer coupled to the database, wherein the server computer includes:

(a) receiving means for receiving an attribute restriction value for restricting a selected attribute; and

(b) segmenting means for segmenting the database, wherein a second heirarchy is created, the second heirarchy being associated with the attribute restriction value; and

(3) a client computer coupled to the server, wherein the client computer includes:

(a) means for inputting from a user of the client computer the attribute restriction value; and

(b) means for transmitting the attribute restriction value to the receiving means.

2. A database management system, characterized by

(1) a database comprising data indicative of an organization;

(2) a server computer coupled to the database, the server computer including:

(a) querying means for querying the database in response to the occurrence of a defined trigger event or in response to a request to analyse the data in the database; and

(b) report generating means for generating a report responsive to the querying means; and

(3) a client computer coupled to the server, the client computer including:

(a) means for receiving the report generated by the report generating means; and

(b) means for displaying the report to a user of the client computer.

3. A database management system comprising:

(1) a database comprising data entities indicative of an organization, the data entities being arranged in a first hierachy with each level of the first heirarchy being associated with at least one attribute;

(2) a server computer coupled to the database, wherein the server computer includes:

(a) querying means for querying the database in response to the occurrence of a defined trigger event or in response to a request to analyse the data in the database;

(b) report generating means for generating a report responsive to the querying means;

(c) receiving means for receiving an attribute restriction value for restricting a selected attribute; and

(b) segmenting means for segmenting the database to create a second heirarchy which is associated with the attribute restriction value; and

(3) a client computer coupled to the server, wherein the client computer includes:

(a) means for inputting from a user of the client computer the attribute restriction value;

(b) means for transmitting the attribute restriction value to the receiving means

(c) means for receiving the report generated by the report generating means; and

(d) means for displaying the report to a user of the client computer.

3. Detailed Description of Invention

The present invention relates to expert systems and reporting systems, and more specifically to a database management system for analysing and segmenting a computer database and for generating reports.

In many applicatons, large amounts of transaction-level data are stored for later analysis (data warehousing). To make use of data warehouses, the data must be retrieved, organized and then presented in an understandable format.

Discovery tools are used to retrieve, analyze and present data from data warehouses. These tools can range from very complex modeling tools to relatively simple end user query tools designed to do no more than mask the complexity of the SQL database programming language from the user. Automated tools that search the data for trends or relationships are also considered discovery tools.

The marketplace is comprised of various tool vendors whose products provide solutions for a portion of the entire knowledge discovery process. Therefore, to effectively utilize their data, the user community is forced to pick multiple, disjoint tools. In addition, these tools are targeted toward the expert user who has an in-depth knowledge of the data and database formats or the various analytic methods that are implemented in the tool. Existing products also do not let the user explicitly and iteratively represent knowledge. Finally, the output of existing tools consists of tables of numbers that users have to analyze and interpret.

Data warehouses, and databases in general, typically have complex structure organized for efficiency of data retrieval, not ease-of-use by the end user. Users desire reports in their vocabulary, not the vocabulary of the database. While some tools in the marketplace allow a user to define new terms and map those terms to the database, the

management of related sets of new terms is not supported. That is, the relationship of a new term to existing terms is not automatically detected for the user.

In addition to these difficulties, it is common for the contents of a report to cause a user to desire another, similar report. Saving and re-using sets of related reports (re-generating the reports over a new set of data) is also desired. The generation of related reports and the re-generation of reports over new data is a capability not adequately available in the marketplace.

It is an object desirable to provide a system for generating reports from a computer database which allow a user to retrieve and analyze data with one tool without requiring the user to have knowledge of underlying data structures or of the database programming language, which allow a user to define new terms and detect and manage relationships between terms, which allow a user to easily generate related reports, and which allow a user to re-run sets of related reports over new data. It is also an object of the present invention to provide a system allowing the a user to segment and partition a database based upon attributes associated with the data in the database.

In accordance with a first aspect of the present invention, there is provided a database management system characterized by:

(1) a database comprising data entities having associated attributes, wherein the data entities are arranged in a first heirarchy, each level of the first heirarchy being associated with at least one attribute;

(2) a server computer coupled to the database, wherein the server computer includes:

(a) receiving means for receiving an attribute restriction value for restricting a selected attribute; and

(b) segmenting means for segmenting the database, wherein a second heirarchy is created, the second heirarchy being associated with the attribute restriction value; and

(3) a client computer coupled to the server, wherein the client computer includes:

(a) means for inputting from a user of the client computer the attribute restriction value; and

(b) means for transmitting the attribute restriction value to the receiving means. the database having the second heirarchy.

According to a second aspect of the present invention, there is provided a database management system, characterized by

(1) a database comprising data indicative of an organization;

(2) a server computer coupled to the database, the server computer including:

(a) querying means for querying the database in response to the occurrence of a defined trigger event or in response to a request to analyse the data in the database; and

(b) report generating means for generating a report responsive to the querying means; and

(3) a client computer coupled to the server, the client computer including:

(a) means for receiving the report generated by the report generating means; and

(b) means for displaying the report to a user of the client computer.

According to a further aspect of the present invention, there is provided a database management system comprising:

(1) a database comprising data entities indicative of an organization, the data entities being arranged in a first hierachy with each level of the first heirarchy being associated with at least one attribute;

(2) a server computer coupled to the database, wherein the server computer includes:

(a) querying means for querying the database in response to the occurrence of a defined trigger event or in response to a request to analyse the data in the database;

(b) report generating means for generating a report responsive to the querying means;

(c) receiving means for receiving an attribute restriction value for restricting a selected attribute; and

(b) segmenting means for segmenting the database to create a second heirarchy which is associated with the attribute restriction value; and

(3) a client computer coupled to the server, wherein the client computer includes:

(a) means for inputting from a user of the client computer the attribute restriction value;

(b) means for transmitting the attribute restriction value to the receiving means

(c) means for receiving the report generated by the report generating means; and

(d) means for displaying the report to a user of the client computer.

Referring now to Fig. 1, system 10 includes four major subsystems: client subsystem 12, data abstraction intelligence (DAI) subsystem 14, data and schema manipulation (DSM) subsystem 16, and scheduler subsystem 18.

In connection with the description of system 10, the following definitions are provided:

An Alert Condition is a user-defined condition or set of conditions that when satisfied returns an Alert Message. For instance, an Alert Condition may be defined so that when the inventory of brand A shirts drops below 200 units for a given week, system 10 produces an Alert Message, InfoFrame™ or runs another analyst.

An Alert Message is a message that notifies the user that an Alert Condition has been satisfied. From an Alert Message the user can select the corresponding Information Report (InfoFrame™) to be run. An example of an Alert Message would be "Alert: the inventory of brand A shirts is below 200."

An Alert Information Report (Alert InfoFrame™) is a type of status report that describes an Alert Message in detail. The Alert InfoFrame™ has a description of what happened, when, and probable reasons why it occurred.

An Analyst specifies an event in the data which must trigger an Alert; or specifies the type of analysis and the measures and segments to be reported on in an InfoFrame™, and optionally the schedule on which this InfoFrame™ is to be generated or the event in the data which must trigger the InfoFrame™.

An attribute is a characteristic or feature of an entity represented in the warehouse. For example, **Color, Manufacturer,** or Size are all attributes of the product category of **Clothing**.

An attribute restriction is an expression that restricts the value that attribute can have. For example, in "Price is less than $10.00", the "less than $10.00" is a restriction on the Prices attribute. Another example might be: "Woman's Clothing or Men's Clothing" is a restriction on the Department Attribute. A single attribute value, like "Blue" or "Men's Clothing", is also an attribute restriction.

A specific entity (like a product) in the data warehouse is represented as a set of attributes and values. For example, the product "Designer X men's shirt, size 42, color blue", might be represented as "Product: Department: Men's Clothing; Manufacturer: Designer X; Size: 42; Color: blue". These values are members of specific domain for each attribute (see below).

Indicators are classifications across Concepts that are usually related to numerical values (e.g. Sales Volume, Inventory, Price). Indicators have methods and formulae that pertain to their computation (e.g. Total Sales) and causal associations between Indicators (e.g. If Price increases Sales Volume should decrease). Within a Indicator, segments can be defined which describe a specific group of Indicators of interest (e.g. Senior Customer, Company A Products).

A Change Analysis Report is a compound document describing Indicators over two time periods. Within system 10, one can specify two periods of time and see the difference of a chosen Indicator for that period (e.g., How did this year's sales of textiles compare to last years sales?) Change Analysis Reports can report results for a day, week, month, quarter, year, or other defined period.

A Comparison Analysis InfoFrame™ is a type of status report which helps a user compare the value of two Indicators across the same time period or compare the value of the same Indicator across two sibling segments across the same time period.

Compound Indicators are user-defined Indicators created by combining primitive Indicators with arithmetic and set operations.

A Data Warehouse is a very large collection of data that is housed in one or more databases. These databases usually reside on database server computers and can be either in one location or distributed geographically.

A Dimension defines the high-level categories of entities. For example, in a Retail domain, the dimensions might be: Product, Market, and Time (Time is a universal dimension applicable to any domain). A dimension has associated with a set of attributes that can be used to describe its entities; for example, **Brand**, **Manufacturer** and Size describe the dimensions of a product..

Every attribute has an implicit or explicit domain of values. For example, the domain of values for the **Department** attribute is an encoding of the legitimate departments for the enterprise, and the domain of values for the Size attribute is a non-zero number representing the size in specified units.

A Drill Down Heuristic specifies some relation between the measure values of the segments of a free attribute of a segment which must be reported to the user.

End Users are users for which system 10 is specifically designed. End users typically have knowledge of a user's operations and for this example have used Microsoft Windows™ (Windows 3.1™, Windows NT™ & Windows 95™, etc.). End users typically do not have expertise in SQL code generation or the specific data structures of the databases they want to access.

Enterprise Information Factory™ (EIF) is a commercial software package that allows typical users to access their data warehouse data. The data warehouse is essentially a passive environment that usually requires the use of SQL code and knowledge about the structure of the database to access data. The EIF differs from the data warehouse by providing a foundation for providing tools to allow users without SQL or database knowledge to get data out of their databases.

An Exception Analyst is specifically an Analyst which runs regularly to test for a trigger condition, and which returns an Alert or a Report when the trigger condition occurs.

If the domain of an attribute is a finite set (like Department above), it is called a finite domain. The alternative is an infinite or continuous domain, like Price.

A Free Attribute is an attribute of a segment which has not been restricted to define that segment. Color, Cost, and Weight might all be free attributes of the segment "Expensive Shirts"

A Heuristic Rule specifies some condition of data, some relation between the segment measure values found by an analyst, that should be reported to the user in the completed InfoFrame™.

HyperText Markup Language (HTML) is an emerging standard format for software documents that allows for the inclusion of hyperlinks and graphics (pictures, graphs, tables) in text documents. A hyperlink is a "hot" area in the document (usually text in a different color than the surrounding text), that when clicked on, shows another document that is related or linked to the original HTML document.

InfoFrame™ Definitions are System Templates that have been customized to include particular Concepts, Indicators, Time Intervals, Analysis Type, and segments. InfoFrame™ Definitions can be immediately "run" to produce a "InfoFrame™", saved to be run later, saved and scheduled to be run later, or triggered by another analyst.

A Knowledge Worker is typically a person familiar with SQL, who knows the structure of the Data Warehouse and who has an analytical background. In addition, the Knowledge Worker may use statistical and data analysis packages and data modeling tools.

Managament Discovery Tool™ (MDT) refers to the overall system of the present invention.

Metadata™ is the collection of information about the end user's particular operation and may be one of three types: core, public or private. After installation this

information is stored in the end user's database and is used to tailor reports to the end user's particular needs. Metadata™ includes, but is not limited to, Concepts, Indicators, Segments, Attributes, Attribute Values and Measure Relationships.

The core set of Metadata™ is typically set up at installation by professional services personnel and the MDT Administrator. Core Metadata™ consists of Dimensions, Attributes, Basic Measures, Segments and Year definitions.

Public Metadata™ is only changeable by the MDT Administrator and Knowledge Worker user types and is defined and modified after installation. Public Metadata™ includes Public Composite Measures, Public Measure Relationships and Public Segments.

Private Metadata™ belongs to each user and is only changeable by the end user (executive user) user type. Private Metadata™ includes Private Composite Measures, Private Measure Relationships and Private Segments.

If an attribute has a finite domain, the Natural Partition is the partition where each segment corresponds to each element of the domain. For example, the natural partition of the Department attribute is the set of segments "Men's Clothing", "Woman's Clothing", etc.

Object Linking and Embedding (OLE) is a computer format that allows objects (e.g., graphs, tables) in computer documents to, when double clicked on, bring up the software application that created the object (graph, table, document).

If the user-defined segments for a given partition do not cover the domain, then an "Other" segment will stand for the rest of the partition.

A partition is an implicit or explicit division of the dimension by the restriction of a single attribute. For example, of one takes the Price attribute, and the "less than $10.00" restriction, this defines a partition of products into two sets or segments: those products with Price less than ten dollars, and those products with Price greater to or equal to ten dollars. Note that the sets or segments of a partition must cover the original set and not overlap, i.e., "Price < $10.00" and "Price < $15.00" do not define a partition.

Primitive Indicators are Indicators that are directly mappable to data in the data warehouse. They are set up during installation of the present invention and are not changeable by the end user.

Reports or InfoFrames™ are compound documents that display data from a database in text and graphics (e.g., graphs, tables). Reports are the result of running a InfoFrame™ Definition. InfoFrames™ may be in the HTML format and may be OLE 2.0 compliant.

A Restricted Attribute is an attribute of a segment which has be restricted to defined the segment. Product and Price might be the restricted attributes of "Expensive Shirts"

Segments are user-defined groups that are defined within a Concept having a meaningful attribute or attributes in common. For instance, the segment "Senior Customers" might be those customers whose age is greater than 65 years.

A segment is one part of a partition. Actually, a segment is a subset of data defined by restrictions on one or several attributes. If a segment is defined by several attributes, it can be part of several partitions, one for each restricted attribute (as shown shortly. This means that, given a segment in isolation, one cannot determine which partition it "should" belong to, and thus, one cannot determine its natural parents in the hierarchy).

A set of segments forms a segment hierarchy under the subset relation, with a root that is the "top-level segment", which contains all of the members of the dimension.

Structured Query Language (SQL) is a structured language for viewing the contents of a relational database.

Summarization InfoFrame™ is a type of InfoFrame™ that shows a roll-up or summarization of a specified Indicator across one or more specified segments. By selecting a particular Indicator in this report a InfoFrame™ showing the "winners" and "losers" for the specified period can be automatically produced.

System Administrators (MDT Administrators, or MDTA) are those users of system 10 who have an intimate knowledge of the databases and data structures of an organization. Often the System Administrator has the title of "database administrator" (DBA).

A Text Generation Rule specifies the text that must be inserted into an InfoFrame™ when the some heuristic rule is satisfied.

A Trend Analysis InfoFrame™ is a type of InfoFrame™ that, when defined, shows the trend for a specific Indicator or indicators over a specified period of time. This analysis can aid in forecasting the future by identifying patterns in past activities.

Client subsystem 12 is a single application program which has a graphical user interface (GUI) 40 and which allows a user to select and specify parameters for InfoFrames™, view InfoFrames™, print InfoFrames™, and save InfoFrames™. Finally, the user can define Composite Indicators and Segments, create Analysts, define Measure Relationships, or modify the schedule of Analysts.

DAI subsystem 14 provides intelligent middleware for translating graphical user requests, selecting system templates, manipulating data views, and generating dimensional queries for retrieving data from data warehouse 24. It also contains rules for choosing default parameters, for choosing layout and display formats, and for generating text from data. DAI subsystem 14 is responsible for instantiating user selected InfoFrames™ and managing several kinds of Metadata™ 25 used in this instantiation. This Metadata™ 25 represents Concepts and Indicators that provide a customizable "dimensionalization" of the relational data in data warehouse 24. DAI subsystem 14 also processes updates to this Metadata™ 25 that originate in client subsystem 12 and handles several other kinds of user updates, primarily by passing them to DSM subsystem 16.

DSM subsystem 16 reads schema from data warehouse 24, creates data views, and creates a mapping between the two. It also uses that mapping to translate the Dimensional Queries received from DAI subsystem 14 into SQL and package and return the results.

Scheduler subsystem 18 is responsible for for starting Analysts which are to run at a scheduled time or on a regulare schedule; or Exception Analsysts which must regularly test for a trigger condition in the database.. When the requested time interval occurs, the Scheduler starts up, requests a list of scheduled InfoFrame™ Requests from DAI subsystem 14. From those lists, scheduler subsystem 18 determines which should be run during the current time interval and sends those requests to DAI subsystem 14 as if they were sent by client subsystem 12.

Thus, system 10 is implemented as a three-tier architecture. Client computer 30 executes client subsystem 12. Client computer 30 preferably executes Windows NT™, or Windows 95™, although other operating systems are also envisioned by the present invention. Client subsystem 12 (Figs. 6-12) is suitable for use with these operating systems. Display 22 and input device 21 allow a user to view GUI 40 and enter choices of Metadata™ 25 used in creating Analysts. Input device 21 may be a keyboard, mouse, or other pointing device. Printer 23 allows a user to print a InfoFrame™. Storage medium 26 allows a user to store an InfoFrame™ or Alert Message.

Server computer 32 executes DAI subsystem 14, DSM subsystem 16, and scheduler subsystem 18. These three subsystems combine to satisfy user requests from client subsystem 12 using information from data warehouse 24. Server computer 32 is preferably a multi-processor computer and executes the UNIX ™ operating system or Windows NT ™, although other computer and operating system configurations are also envisioned by the present invention.

Client and server computers 30 and 32 are preferably coupled asynchronously for report requests; all other requests are satisfied synchronously. Communication between client and server computers 30 and 32 is preferably through transmission control protocol/ internet protocol (TCP/IP), although other transmission protocols are also envisioned by the present invention.

Database computer 34 includes one or more storage media 36 containing data warehouse 24. Database computer 34 is preferably a massively parallel processor

computer and executes the UNIX ™ operating system or Windows NT ™, although other computer and operating system configurations are also envisioned by the present invention. Data warehouse 24 is suited to run on any computer which supports an Open Database Connect (ODBC) interface to data warehouse 24. Communication between server computer 32 and database computer 34 is preferably via ODBC, although other database interfaces are also envisioned by the present invention.

Turning now to Fig. 2, client subsystem 12 is an application program which gives a user control over system 10 and is suitable for execution on top of the Windows NT ™, or Windows 95 ™ operating systems. Client subsystem 12 includes log-in module 50, folder management subsystem 54, segment builder 55A, measure builder 55B and measure relationship builder 55C, analyst definition subsystem 56, InfoFrame™ viewing subsystem 53 and MDT Administrator interface 57.

Log-in module 50 verifies that only one copy of the client subsystem 12 is running on computer 30, checks the localization of computer 30, connects to computer 32, and interacts with the user to log him onto client subsystem 12. During logon, log-in module 50 verifies a user's name and password and then retrieves any user preferences that may have been earlier defined. The only request from a user that is handled by log-in module 50 is a request to log onto client subsystem 12.

Log-in module 50 issues the following requests:

• single program running	to Operating System (DOS ™, NT ™, Windows 95 ™, etc.)
• retrieve localization	to Operating System (DOS ™, NT ™ Windows 95 ™, etc.)
• connect to server	to Client/Server module
• disconnect from server	to Client/Server module
• authenticate user	to Metadata™ API 60
• run main menu	to Main Menu 51
• run admin menu	to MDT Administrator Interface 57

If the user is the System Administrator, log-in module 50 displays MDT Administrator interface 57 "Setup" menu item. If the user is an end user or knowledge worker, a Main menu and toolbar interface 51 are displayed, as are the interfaces associated with subsystems 53-55.

MDT Administrator interface 57 is used by a System Administrator to perform system administration tasks, such as making user-defined segments available globally and creating and editing Concepts. Interface 62 is preferably only available to System Administrators during system installation.

Folder management subsystem 54 handles all functions related to manipulating, storing, and retrieving Folder hierarchies, and the InfoFrames™ and Agents that are stored in those Folders. It also handles querying from DAI subsystem 14 for newly-completed InfoFrames™, both when client subsystem 12 starts up, and then periodically thereafter.

Folder management subsystem 54 also handles User requests for operations on:

- Folders (new, delete, rename)
- Agents (edit, delete, run now, print)
- InfoFrames™ (view, delete, annotate, print [in cooperation with the InfoFrame™ View Window])

Each folder is represented by one folder object. A folder stores a list of child folders, a list of InfoFrames™, and a list of Agents. Folder objects are created and deleted by folder management subsystem 54 in response to user requests.

Subsystems 55B provides a user with the ability to create new measures, update measures, or delete existing measures. This information is sent to a Metadata™ API 60 and thereafter to DAI subsystem 14 for updating the user's Metadata™ 25.

Subsystem 55A provides a user with the ability to create new Segments, update segments, or delete existing Segments. This information is sent to a Metadata™ API 60 and thereafter to DAI subsystem 14 for updating the user's Metadata™ 25.

Finally, Subsystem 55C provides a user with an interface to modify measure relations and to constrain measure relations. The user selects the current measure and whether to evaluate that measure's relationships when it increases or decrease. Then the user can then select from a list of other measures and define their relationship to the current measure. These relationships are in the form of "decreases", "increases", or "is unrelated to the current measure". Also, every relationship between two measures can be constrained. The relationship between measures and the constraints placed upon them are saved on computer 32 for use in generating InfoFrames™.

Analyst definition subsystem 56 handles all functions related to user selection of parameters needed to generate specific reports. It also allows the user to define and schedule Alerts for scheduled reports.

The user may invoke an existing Analyst, delete one from within the folder management subsystem 54, or create a new Analyst. The five types of Analysts are:

- Summarization
- Segment Comparison
- Measure Comparison
- Change Analysis
- Trend Analysis

The Summarization Analyst requires the following user selection requirements:

- Analyst name
- Primary measure, other optional measures
- Primary segment, other segments
- Time period

- Optional schedule
- Optional trigger
- type of year used
- optional trigger event (Alert Message, InfoFrame™, Run another analyst)

The Segment Comparison Analyst requires the following user selection requirements:

- Analyst name
- Primary measure
- Primary segment, a comparison segment
- Time period
- Optional schedule
- Optional trigger
- type of year used
- optional trigger event (Alert Message, InfoFrame™, Run another analyst)

The Measure Comparison Analyst requires the following user selection requirements:

- Analyst name
- Primary measure, Comparison measure
- Primary segment, other optional segments
- base time period, comparison time period
- Optional schedule
- Optional trigger
- type of year used
- optional trigger event (Alert Message, InfoFrame™, Run another Analyst)

The Change Analysis Analyst requires the following user selection requirements:

- Analyst name
- Primary measure
- Primary segment, Other optional segments
- base time period, comparison time period
- Optional schedule
- Optional trigger
- type of year used
- optional trigger event (Alert Message, InfoFrame™, Run another Analyst)

The Trend Analysis Analyst requires the following user selection requirements:

- Analyst name
- Primary measure
- Primary segment, other optional segments.
- Time period, Time interval.
- Optional schedule
- Optional trigger
- type of year used
- optional trigger event (Alert Message, InfoFrame™, Run another Analyst)

The user can save or run the analyst definition. The user is restricted to choosing one Segment from within each Concept with the exception of Target Segment, in which case he may select only one segment and more than one child partition of the selected segment. The user may choose to schedule an Analyst or to modify or delete an existing schedule. Unscheduled Analysts will be run when the user commands. Scheduled Analysts will be submitted to the server for execution at a later date or periodic execution.

The user may specify a trigger condition for the Analyst to specify an Exception Analyst. When submitted to the server it will be run regularly to test for its trigger condition, and will return an Alert or an InfoFrame™ whenever the trigger condition occurs.

The Analyst definition subsystem 56 makes the following requests to the folder management subsystem 54:

Save	Check if the user has selected the appropriate parameters for the selected analyst. Send a request to the folder management subsystem 54 to save an existing Analyst Definition
Save As	Check if the user has selected the appropriate parameters for the selected analyst. Send a request to the folder management subsystem 54 to save an existing Analyst Definition
Submit	Check if the user has selected the appropriate parameters for the selected analyst. Send a request to the folder management subsystem 54 to submit a report generation

The Analyst definition subsystem also makes the following requests to Metadata™ API 60:

Get all Measures	The request will be made to Metadata™ API 60 each time there is a need for it at the initialization point of a dialog
Get all Concepts	The request will be made to Metadata™ API 60 subsystem each time there is a need for it at the initialization point of a dialog
Get a Concept's Partitions	The request will be made depending on a user's selection of a concept
Get Partitions	The request will be made depending on a

	user selection of a defined Segment.
Get Segments	The request will be made depending on a user selection of a partition.

InfoFrame™ viewing subsystem 53 includes a WYSIWYG browser which displays a selected InfoFrame™ on screen, when InfoFrame™ viewing subsystem 53 gets a notification from folder management subsystem 54 to view a InfoFrame™. If the user decides to drill down from the current InfoFrame™, InfoFrame™ viewing subsystem 53 notifies the folder management subsystem 54 to send a new report request.

When the user double-clicks on an InfoFrame™ or chooses "menu item - View" from the File menu Folders, the folder management subsystem 54 notifies the InfoFrame™ viewing subsystem to view the InfoFrame™. When the user clicks on a hypertext to drill down from the current InfoFrame™, the InfoFrame™ viewing subsystem 53 passes the drill down information to the folder management subsystem 54 to send a new report request to DAI subsystem 14.

InfoFrame™ viewing subsystem 53 includes a parser which parses the InfoFrame™, and extracts the completed report, which is written in HTML. In an HTML file, HTML tags indicate document elements, structure, formatting, and hypertext linking to other documents or resources. The parser then outputs all the information for display. In the current invention, the hyperlink may instance a new Analyst and a new InfoFrame™

The InfoFrame™ viewing subsystem 53 allows a user to display and format text, tables, and graphs displayed by display 22 based on the information gathered by the parser. A header, a footer, and annotations can be added to a InfoFrame™. The user can save the viewed InfoFrame™. The user can also save an InfoFrame™ as a HTML file in either UNICODE or ASCII code format. A saved HTML InfoFrame™ can be attached to an e-mail to mail out. Any HTML version 3.0 browser, or equivalent, can read the HTML InfoFrame™.

Metadata™ API 60 handles most of the communications between client subsystem 12 and DAI subsystem 14. These communications involve four basic types of

data: Metadata™ 25, InfoFrames™, user profiles, and data warehouse schema. For Metadata™ communication, Metadata™ API 60 provides the ability to add, delete and update Metadata™ 25. For InfoFrames™, Metadata™ API 60 provides the ability to request a report, get the status of a report, retrieve a report and cancel a report request. For user profiles, Metadata™ API 60 provides the ability to add a user, authenticate a user and delete a user. The communication for data warehouse schema is to retrieve it.

Metadata™ API 60 allows a user to define new ways of looking at an operation. A user cannot modify the public segments, the basic measures or the public measures. However, the user can create new Indicators and new Segments. In a typical organization of users and system administrators, only system administrators can create or change basic operation measures. Administrators and knowledge workers can create, edit or delete public composite measures, public segments and public measure relationships.

The Metadata™ API 60 will handle the following requests from other client subsystems:

update Metadata™	from subsystems 55A/55B/55C
get report status	from Folder management subsystem 54
generate report	from Folder management subsystem 54
retrieve report	from Folder management subsystem 54
retrieve schema	from MDT Administrator Interface 57
update schedule	from Analyst Definition subsystem 56
cancel a report	from Analyst Definition subsystem 56
authenticate user	from Log-in module 50
add a user	from MDT Administrator Interface 57
delete a user	from MDT Administrator Interface 57
update user password	from MDT Administrator Interface 57

Metadata™ API 60 sends the following requests directly to DAI subsystem 14:

- disconnect from computer 32
- send data to DAI subsystem 14

- receive data from DAI subsystem 14

Turning now to Fig. 3, DAI subsystem 14 includes return area manager 70, InfoFrame™ generator 72, Metadata™ request module 74, Metadata™ repository 76, and Metadata™ load and update module 78.

Metadata™ repository 76 contains a representation of Metadata™ 25 within data warehouse 24. This Metadata™ 25 is the core of system 10; it provides a customizable view over the relational data in warehouse 24 and is the primary vocabulary for the specification of InfoFrames™. Metadata™ repository 76 gets populated at startup time by DSM subsystem 16 from the persistent Metadata™ representation in data warehouse 24.

There are four fundamental kinds of Metadata™ 25 in Metadata™ repository 76, listed and described below:

- Concepts: concepts represent the dimensions along which the data can be viewed. Each dimension imposes a hierarchy over the underlying data, and dimensions can be combined to drive "drill-down" or "drill-up" operations. For example, a simple retail application might have two Concepts: Market and Product. The Market hierarchy is composed of Sales Regions, each of which consists of several States, each of which consists of a set of Stores. The Product Hierarchy is composed of a set of Departments (Home Electronics, Men's Clothing, Hardware), each Department is composed of product Categories (Shirts, Shoes, Slacks), and each Category is composed of individual manufacturer's product lines. Time is a dimension that is important in all applications, and will be represented in system 10. Users can add new Concepts (see below). These, as all of the Metadata™ 25 in Metadata™ repository 76, must be mapped into relational form (that is, into SQL) in order to actually query data warehouse 24. Mapping is done by DSM subsystem 16 during the process of processing Dimensional Queries (see below).
- Indicators: Indicators are the important measures of data of interest. For example, product Volume, Price, or Current Stock are all Indicators. The use of time in a query

further refines the idea of a Indicator; for example, "Change in Volume" applies between two time periods.

- Alerts: Alerts are essentially tests over the data, but they are not part of the Metadata™. They are specified in the Analyst in terms of the Metadata™. For example, a user might specify that if the available stock of a product falls by some percentage, to generate the appropriate InfoFrame™. The user also specifies how often to check the Trigger condition. A list of Alerts is maintained by DAI subsystem 14 and executed by scheduler subsystem 18. This Metadata™ 25 is also available to DAI subsystem 14 and is used to generate InfoFrame™ information.
- Measure Relationships: Measure Relationships are simple expressions of causality; for example, "Increased Sales mean Increased Profit". This kind of Metadata™ 25 is used to generate supporting information for a InfoFrame™ or, alternatively, alert the user to trends that run counter to the set of Measure Relationships.

Metadata™ 25 is initially created during installation of the present invention at the customer's site. The process of creating the Metadata™ 25 is illustrated in more detail in Figs. 7A-7E. What is included within Metadata™ 25 depends on the industry (some Metadata™ 25 will be industry-specific and usable by all companies in that industry), the specific customer of the present invention, and the structure of the customer's data warehouse 24. During installation, some industry-specific Metadata™ 25 is used, some company specific Metadata™ 25 may be created, and the mapping information needed to map Metadata™ 25 to data warehouse 24 is created. All Metadata™ 25, including the mapping information, is stored in a set of relational tables. These relational tables are kept in data warehouse 24 and used by the present invention to create reports for the user.

Metadata™ request module 74 handles all requests for Metadata™ 25, either from client subsystem 12 or DAI subsystem 14. Client subsystem 12 requests Metadata™ 25 from DAI subsystem 14 to be presented to the end users. InfoFrame™ generator 72 requests Metadata™ 25 in order to create Dimensional Queries as part of instantiating a

InfoFrame™ for a user. A request for Metadata™ 25 might be, for example, a request for all sub-concepts of a particular Concept.

Metadata™ request module 74 also handles Metadata™ updates from client subsystem 12. A user adds new Segments by specifying a new dimension from which to group the data. This dimension must be supported by an existing data attribute in the warehouse data. For example, a Product may include a List-Price and a Discount-Price. The user can specify a new dimension called "Discount-Factor", specified using the percent difference between the Discount-Price and the List-Price, and use that to create three new Segments: Heavily-Discounted Products, Slightly-Discounted Products, and Non-Discounted Products. These new Segments can now be used in subsequent InfoFrame™ requests, and, if indicated by the user, made persistent by writing them back into data warehouse 24 by Metadata™ load and update module 78.

Request Structures are passed from one subsystem to another when one subsystem requires processing and results from another. Request Structures vary according to the type of request being sent. Most requests, however, have some common attributes, such as an identification field, an owner, a name and a description of the request.

Concept Update Requests are sent from client subsystem 12 to DAI subsystem 14 and are preferably issued only by the System Administrator. Concept Update Requests are requests for adding a new Concept to the Metadata™ 25. The requests have the following format:

BC_ID:	ID which uniquely identifies this Concept
BC_NAME:	The name of this Concept
BC_DESC:	The description of this Concept
MAPPING:	Mapping of this Concept to data warehouse tables

Indicator Update Requests are sent from client subsystem 12 to DAI subsystem 14. Indicator Update Requests are requests for adding a new Indicator to the Metadata™ 25.

Indicator Update Requests primarily include primitive and compound requests. Primitive requests have the following format:

BI_ID:	ID which uniquely identifies this Indicator
OWNER:	The user who created this Indicator
BI_NAME:	The name of this Indicator
BI_DESC:	The description of this Indicator
MAPPING:	Mapping of this Indicator to data warehouse tables
ROLLUP_OP:	Operator for performing the roll-up operation

Compound requests have the following format:

BI_ID:	ID which uniquely identifies this Indicator
BI_NAME:	The name of this Indicator
BI_DESC:	The description of this Indicator
EXP:	The expression which describes this Indicator function

Causal Indicator Update Requests are sent from client subsystem 12 to DAI subsystem 14. Causal Indicator Update Requests add a new Causal Indicator to the Metadata™ 25. The request has the following format:

CI_ID:	ID which uniquely identifies this Casual Indicator
OWNER:	The user who created this Causal Indicator
CI_NAME:	The name of this Causal Indicator
CI_DESC:	The description of this Causal Indicator
BI_ID1:	Indicator which is the independent variable of this causal relationship
OP:	The operator for this causal relationship
BI_ID2:	Indicator which is the dependent variable of this causal relationship
RANGE:	When OP is +/-, the range where it is + and the range where it is -

Schema Requests are sent from client subsystem 12 to DAI subsystem 14 and may only be issued by the System Administrator. Schema Requests are requests to retrieve the data base schema from data warehouse 24. This type of request is just a simple unformatted message to DAI subsystem 14.

Segment Update Requests are sent from client subsystem 12 to DAI subsystem 14. Segment Update Requests are requests for adding a new Segment to the Metadata™ 25. Segment Update Requests have the following format:

SEG_ID:	ID which uniquely identifies this Segment
OWNER:	The user who created this Segment
SEG_NAME:	The name of this Segment
SEG_DESC:	The description of this Segment
SEG_LEVEL:	Level in the Segment Hierarchy of this Segment
BC_ID:	The Concept for this Segment
ATTR_ID:	The Attribute(s) for this Segment
OP:	The operator(s) for this Segment
VALUE:	The value (s) for this Segment

InfoFrame™ Requests are sent from the Client subsystem to the DAI subsystem. This type of request is to create a new InfoFrame™ based on user specified selections. The request has the following format:

SR_ID:	ID which uniquely identifies this InfoFrame™
OWNER:	The user who created this InfoFrame™
SR_NAME:	The name of this InfoFrame™
SR_DESC:	The description of this InfoFrame™
SR_TYPE:	One of the four types of InfoFrames™
BC_ID:	The Concept for this InfoFrame™
SEG_ID:	The Segment(s) for this InfoFrame™
TIME:	The time interval(s) for this InfoFrame™

Dimensional Queries are sent from DAI subsystem 14 to DSM subsystem 16. Dimensional Queries formulate requests for data from data warehouse 24. DSM subsystem 16 converts Dimensional Queries into SQL statements.

The DAI subsystem 14 communicates a dimensional query to the DSM subsystem 16 as a list of Metadata™ segment definitions or partition definitions, a list of Metadata™ measure definitions and a Measure Value Table. The DSM subsystem 16 converts these to SQL Queries and submits them to the Data Warehouse 24. The results returned by the Data Warehouse to the DSM are returned to the DAI in the Measure Value Table.

Client subsystem 12 produces the following outputs to DAI subsystem 14:

- Concept Update Requests
- Indicator Update Requests
- Causal Indicator Update Requests
- Schema Requests
- Segment Update Requests
- InfoFrame™ Requests
- Cancel Requests

DAI subsystem 14 provides the following outputs to client subsystem 12:

- Concept Structures
- Indicator Structures
- Causal Indicator Structures
- Schema Structures
- Segment Structures
- InfoFrames™
- Error/Status Codes

DAI subsystem 14 provides the following outputs to scheduler subsystem 18:

- Schedule Analyst Request
- Delete Analyst Request

DAI subsystem 14 provides the following outputs to DSM subsystem 16:

- Dimensional Queries
- Metadata™ Retrieval Requests
- Schema Requests

DSM subsystem 16 provides the following outputs to DAI subsystem 14:

- Updated Metadata™
- Data from the Data Warehouse
- Database Schema

DSM subsystem 16 provides the following outputs to data warehouse 24:

- SQL Statements

DSM subsystem 16 receives the following inputs from data warehouse 24:

- Metadata™
- Database Schema
- Warehouse Data

Scheduler 18 provides the following output to DAI subsystem 14:

- Analyst Definitions

Metadata™ load and update module 78 populates Metadata™ repository 76 from the persistent Metadata™ stored in data warehouse 24 upon system startup. In addition, when a user specifies new Concepts and indicates that he wants them saved,

Metadata™ load and update module 78 writes them back into data warehouse 24 for future use.

InfoFrame™ generator 72 fulfills the primary purpose of DAI subsystem 14. Report generation begins when a user's Analyst containing an InfoFrame™ definition is received by the DAI. The type of Analyst is used to select appropriate Drill Down Heuristics and Text Generation Rules from the set implemented in the DAI. Drill Down Heuristics are used to determine if there any data relationships between the segments of the free attributes of the target segment which must be reported. Text Generation Rules are used to determine what features of the target segment ought to be reported and what relationships to sibling segments, other segments in the restricted attributes of the target segment, ought to be reported. Text Generation rules may specify localizable text, graphs or tables as appropriate output. The output of the Report Generation process is a fully instantiated InfoFrame™ returned to client subsystem 12 in the form of HyperText Markup Language (HTML), a widely-used standard for building portable compound documents.

InfoFrame™ generator 72 has several kinds of knowledge:

- Knowledge of how to map Abstract Queries into Dimensional Queries
- Knowledge of how to use Metadata™ 25 to generate default choices (choices not made by the user in the InfoFrame™ Request)
- Knowledge of how to use both Metadata™ 25 and data returned from the warehouse to guide the selection of both text components
- Knowledge of how to use both Metadata™ 25 and data returned from the warehouse to guide the selection of different types of graphical presentations.

For example, the Summary InfoFrame™ may take as arguments a Concept, a Indicator, and a time period. The Report Generation Module uses the user selected parameters, for example, the Concept "Product", the Concept Segment "Men's Shirts", the Indicator "Volume", and the time period "December 1994" to create a Dimensional

Query. This Dimensional Query is sent to the Data and Schema Manipulation subsystem, which translates this query into SQL and actually executes it. It returns the computed data to DAI subsystem 14, where other Abstract Queries might embed the actual number in a bullet.

Other Abstract Queries have conditionals associated with them. To build off the previous example, another part of the summary System Template might specify the creation of a graph, showing how the target-business-indicator (volume) is apportioned among the segments of the target-business-concept (shirts). In this case, report generator 72 makes a Metadata™ request to return the set of segments, in this example, the dimension that specifies the shirt manufacturer. All volume information is requested for each manufacturer of shirts. Now, additional information guides report generator 72 in the selection of a choice of graph. For example, if the number of segments (manufacturers in this case) is small, like 7 or less, then a pie graph is appropriate, otherwise, a bar graph is preferred. If the number of segments is very large, then aggregate the bottom 20 percent (in terms of the Indicator, in this case, Volume) and use that aggregate with the label "Other" in the graph.

Return area manager 70 keeps track of InfoFrames™ and Alert Evaluations with positive results by user that are waiting for delivery to client subsystem 12. When a user logs into system 10, client subsystem 12 issues a request to DAI subsystem 14 to return all data for that user in the return area. Return area manager 70 retrieves the information from the return area on server computer 32 and sends it back to client computer 30 through DAI subsystem 14.

Turning now to Fig. 4, DSM subsystem 16 includes SQL generator 80 and Metadata™ query module 82.

SQL generator 80 translates dimensional queries received from DAI subsystem 14 into SQL statements used to retrieve data from data warehouse 24. A mapping from concepts to database entities is stored in the Metadata™ 25 and is used in the

formatting of the SQL statements. SQL generator 80 provides to DAI subsystem 14 for use in creating InfoFrames™.

Metadata™ query generator 82 processes requests for Metadata™ 25 submitted by DAI subsystem 14. At system startup, DAI subsystem 14 requests all Metadata™ 25 in order to initialize the knowledge base. Metadata™ query generator 82 is also invoked whenever the user modifies his Segments, causing DAI subsystem 14 to issue an update Metadata™ request.

Turning now to Fig. 5, scheduler subsystem 18 includes alert and report scheduler 90. The scheduler periodically tests queued Scheduled Analysts and will dispatch those to the have come due to the DAI subsystem 14. It will periodically dispatch all submitted Exception Analysts to the DAI subsystem 14 so that they can test for trigger conditions. The schedule and trigger periods are independently configurable by the MDT Administrator. The scheduler passes analysts to the CDAI 14B, by way of the Dispatcher 2513 (Fig. 27).

Turning now to Figs. 6-12, client subsystem 12 and its operation are illustrated in more detail.

Client subsystem 12 includes a primary overlay 98 which appears when client subsystem 12 is executed. Overlay 98 includes three display areas 100-104 within a common Folders window, pull-down menus 106, and buttons 110-120. The Folders window may be maximized (as it is shown in Fig. 6) to eliminate its borders, resized, or minimized as an icon within client subsystem 12. The Folders window cannot be closed.

Display area 100 contains a list of folders, which represent the metaphor used by client subsystem 12 in organizing InfoFrames™ and the analysis that creates them. A folder is opened by highlighting it and selecting it with input device 21. The first folder in the list is opened by default when client subsystem 12 is executed.

Display area 102 contains a list of InfoFrames™ within a selected folder. A InfoFrame™ may be viewed by highlighting it and selecting it with input device 21. An Analysis window 136 appears containing the InfoFrame™. The title bar of the window

indicates the type of preselected analysis that has been performed. For example, in Fig. 12, "change" analysis was preselected by a user to be the type of analysis to run. The Analysis window 136 may be maximized (as it is shown in Fig. 12) to eliminate its borders, resized, or minimized as an icon within client subsystem 12. The Analysis window 136 may be closed by selecting button 122 (Fig. 12) or by a manner well known to users of Windows 3.1™, Windows 95™, and other windows operating environments.

Display area 104 contains a list of Analysts within a selected folder. An Analyst is a personification of preselected operations performed on preselected data for the purpose of generating a InfoFrame™. An Analyst may be viewed by highlighting it and selecting it with input device 21. Analyst Builder windows 130 (Figs. 7A-7E) appears containing the preselected settings saved within the Analyst and used to generate the corresponding InfoFrame™ listed in display area 102. (The InfoFrames™ listed in display area 102 are arranged in the same order as the Analysts listed in display area 104, and have the same titles as the corresponding Analysts). The Analyst Builder window 130 may be not be maximized, resized, or minimized as an icon; it may only be closed in a manner well known to users of Windows 3.1™, Windows 95™, and other windows operating environments.

Buttons 110-122 (Fig. 6) implement the primary operational commands within pull-down menus 106 and are activated using a pointing device. Button 110 calls the Analyst Builder windows 130 (Figs. 7A-7E).

Button 112 calls a Segments divider within a Information Setup window 132 (Fig. 8A). Button 116 deletes a selected file or folder within display areas 100-104. Button 118 creates a new folder. Button 120 calls the Analysis window 136 with a selected InfoFrame™ from display area 102. Button 122 closes client subsystem 12. Button 150 is a print button, button 151 allows the user to create measures, and button 152 allows the user to create or edit measure relationships.

With reference to Figs. 7A-7E, Analyst Builder window 130 allows a user to define how selected data is analyzed. An Analyst is named under the Analyst Name

field. A type of analysis is chosen under the Type of Analysis field. A primary measure to be used in implementing the analysis is chosen under the Primary Measure field. Segments to be reported on are chosen from the list of Defined Segments. Finally, a period for the InfoFrame™ is defined under the Time Slice Considered fields. A InfoFrame™ can be created immediately by selecting the Report Now button, or can be scheduled as part of a batch of InfoFrames™ by selecting the Schedule Analyst button.

With reference to Fig. 8A, the Segments divider within the Information Setup window 132 allows Segments to be created, modified, or deleted. A description of the segment appears in the Description field. Upon activation of button 801 by the user, the window 132 of Fig. 8B is launched, allowing the user to edit segment definitons.

With reference to Fig. 9A, Measures of information may be created and modified within the Measures divider of the Information Setup window 132. A name for each Measure appears in the Measure Name field. A definition of a Measure appears in the Definition field. Mathematical operators, Time Slice constraints, Segment constraints, and constraints from other Measures may be inserted into the Definition using the corresponding buttons below the Definition field. With respect to Figs. 9B and 9C, windows 132 may be displayed to select measures and select segments, respectively.

With reference to Fig. 10, Measure relationships may be defined and modified within the Measure Relations divider of the Information Setup window 132. Measure relationships are defined in terms of an if-then statement. A primary measure and whether it increases or decreases is selected in the Measure field, which represents the "If" part of the If-Then statement. Measures within the Unrelated field may be moved to either the Decreases field or the Increases field to form the "Then" part of the If-Then statement. With respect to Fig. 11, measure relationships may be restricted by means of the window 132 of that figure.

A batch of InfoFrames™ may be individually scheduled for automatic production. Scheduling of InfoFrames™ is particularly useful to those users that require

periodic InfoFrames™. InfoFrame™ time intervals may be selected under the Time Interval field, which provides daily, weekly, and monthly reporting options.

With reference to Fig. 12, a sample InfoFrame™ is shown within Analysis window 136. The type of analysis performed is indicated in the InfoFrame™ and in the title bar as "Change Analysis". The Segment (previously defined within the Segments divider of the Information Setup window 132) is "Store Ages Greater than 3 Years". The Measure (previously define within the Measures divider of the Information Setup window 132) is "Same Store Sales". The Time Slice (previously defined in the Time Slice Considered fields of the Analyst Builder window 130) is "Year to date 1995 vs. Last Year".

The InfoFrame™ provides a concise statement of changes that have occurred in the Primary Measure, Same Store Sales, and changes that have occurred in Measures related to the Same Store Sales, Stores Remodeled, and previously defined within the Measure Relations divider of the Information Setup window 132. The InfoFrame™ then contains an explanation, including a graph, for the change in the Primary Measure, Same Store Sales.

InfoFrame™ may include multiple instances of HTML associated with a Measure, representing hyperlinks to text data or graphic data representing the results of the Measure.

Turning now to Fig. 13, a method for creating Metadata™ 25 using client subsystem 12 is illustrated beginning with START 140.

In step 141, the user specifies a Concept.

In step 142, the user specifies one or more attributes for the Concept.

In step 144, client subsystem 12 provides the user with the list of columns of tables in data warehouse 24.

In step 146, the user maps every attribute to a column. The user can provide a textual description of the concepts and the attributes.

In step 148, the user specifies one or more indicators by "mapping" a Indicator to a column in a table within data warehouse 24.

In step 150, client subsystem 12 provides the user with a list of columns for the purpose of mapping Indicators as well.

In step 152, user selects an "aggregate method" for the Indicator that is mapped, which specifies how values for the Indicator are aggregated. The system supports the following aggregate methods:

- Add
- Average
- Min
- Max
- Count
- Last in period
- First in period

In step 154, the user selects the unit of measurement, and specifies whether the Indicator is a currency. The user can optionally specify a plural form of the Indicator, a verb to describe change in the value of the Indicator, the precision for reporting the Indicator and a textual description of the Indicator.

In step 156, client subsystem 12 ensures that tables having Indicator columns can be joined with tables that have Attribute columns.

In step 158, client subsystem 12 determines whether the user wishes to enter additional Concepts. If so, the method returns to step 142. If not, the method ends at step 160.

The preceding description forms an overview of the present invention. The following sections describe the invention in further detail, broken into further sections.

2. Client Subsystem 12

The client subsystem 12 is described in further detail below.

Fig. 14 illustrates a more detailed block diagram of the client subsystem 12. Client subsystem 12 contains three subsystems: User Interface (UI) 1401, Manager 1402, and Server APIs 1403. As its name implies, the user interface subsystem 1401 allows the user 1405 to interact with the client 12. At this level of detail, it can be seen that the User Interface subsystem 1401 uses services of both the Manager 1402 and the Server APIs 1403; the Manager 1402 also uses services from the Server APIs 1403. The Server APIs subsystem 1403 provides high level APIs which abstract all client 12 interactions with the DAI subsystem 14. All communications between the client 12 and the DAI subsystem 14 are sent through the Client Server Module (CSM) 1404, which is described in further detail below.

Fig. 15 illustrates a block diagram of the client subsystem 12 having an increased level of detail over the block diagram of Fig. 14. The user interface subsystem 1401 contains all portions of the program that are visible to the user 1405. Because this subsystem may be implemented as a standard MS-Windows style program, most of the units within the interface are either windows or dialog boxes. Each window or dialog box in the interface has one main class which defines its behavior, as detailed below. Some window or dialog classes also use other utility classes, which will also be defined below, where appropriate.

The "top level" of control within the client subsystem 12 is the Application object 1511. The application object 1511 is constructed automatically by the Microsoft Foundation Class (MFC) library's start-up code. The application object has two primary responsibilities: performing login validation, and displaying the main frame window. The frame window in a Multiple Document Interface (MDI) application owns the Menus, Toolbar, and Statusbar, and creates child window objects.

The User Login process consists of two steps: getting a User Name and Password from the User, and sending them to the Connect function of the Server APIs

subsystem 1403. There are four possible results from an attempted Connect to the server 32:

- login succeeded
- login failed
- too many login failures
- no response from server 32; network down

Upon an unsuccessful login, the login dialog is re-displayed, and the user 1405 may re-enter his/her name and/or password. After a certain number of unsuccessful attempts (number determined by server 32, not client 12), the server 32 will return the "too many failures" result, and the client 12 program will inform the user 1405 of this result, and then exit. If the network or server are down, the client 12 will start up in "off-line" mode, which allows the user 1405 to view saved InfoFrames™, but not to create or edit Analysts, or send InfoFrame™ generation requests.

Upon a successful connect, the application will display the main frame window. A successful Connect result additionally returns an indication of whether the user 1405 has Administrator (MDTA) privileges; if so, the frame window is informed, so that special menu items may be enabled.

The Applicatioon object 1511 may make the following requests of other subsystems:

Function Used	Subsystem
Connect	Session Management API *[Server APIs subsystem 1403]*
Disconnect	Session Management API *[Server APIs subsystem 1403]*
Display ManagerWindow	Manager Window *[UI subsystem 1401]*

The Application object 1511 is an instance of the clnt_App class. It creates one instance each of clnt_UserLoginDlg and clnt_MainFrame.

Class clnt_App is a subclass of the MFC class CWinApp. We inherit most of the standard behavior of the CWinApp, but override the InitInstance function, in which we run the User Login process, and if successful, construct our main window, an instance of clnt_MainFrame.

clnt_MainFrame is a subclass of MFC class CMDIFrameWnd. We override the OnCreate function, in order to initialize the Toolbar, and Menus, and to create the initial Manager Window instance 1512.

The clnt_MainFrame instance handles some of the Menu and Toolbar requests, while others are handled by whichever Child Window is active (one of the four Manager Windows 1512 or the InfoFrame™ Viewer window 1517, as described below). The clnt_MainFrame instance is also responsible for enabling/disabling menu items that vary depending on which Child Window is active.

The User Login dialog is controlled by an instance of the clnt_UserLoginDlg class, a subclass of the MFC class CDialog. The clnt_UserLoginDlg instance displays a dialog which asks the User to enter a name and password. The name and password strings are returned to the calling function when the User clicks the "OK" button.

The Toolbar is controlled by an instance of class clnt_Toolbar, a subclass of MFC's CToolBar. Class clnt_Toolbar inherits all functionality from CToolBar, and adds support for drag-and-drop. Instances of clnt_Toolbar accept drops of one Folder (onto Trash button), one or more Analysts (onto Trash, RunNow, or View buttons), and one or more InfoFrames™ (onto Trash, View, and Print buttons).

The Information Definition 1515 includes all functionality related to addition, modification or deletion of Segments, Measures, and Measure Relationships.

Three dialog boxes are used in the Information Definition 1515 process; one for each type of Information to be edited. The dialogs are controlled by instances of the following classes which are instantiated by clnt_MainFrame (in response to User requests through the Menu or Toolbar).

Clnt_BuildMeasureDlg. This dialog allows the user to update or delete an exisiting measure, or create a new Measure.

Clnt_BuildSegmentDlg. This dialog allows the user to update or delete an exisiting segment, or create a new segment by defining attribute restrictions.

Clnt_BuildRelationDlg. This dialog allows the user to update or delete an exisiting MeasureRelation, or define a new relationship.

The Measure Relationship dialog uses the following services from other subsystems:

Function Used	SubSystem
GetMeasureRelationship	Metadata™ API *[Server APIs subsystem 1403]*
AddMeasureRelationship	Metadata™ API *[Server APIs subsystem 1403]*
DeleteMeasureRelationship	Metadata™ API *[Server APIs subsystem 1403]*
UpdateMeasureRelationship	Metadata™ API *[Server APIs subsystem 1403]*

The Measure dialog uses the following services:

Function Used	SubSystem
AddMeasure	Metadata™ API *[Server APIs subsystem 1403]*
DeleteMeasure	Metadata™ API *[Server APIs subsystem 1403]*
UpdateMeasure	Metadata™ API *[Server APIs subsystem 1403]*
GetAllAnalysts	Manager API *[Manager APIs subsystem 1402]*

The Segment dialog uses the following services:

Function Used	SubSystem
AddSegment	Metadata™ API *[Server APIs subsystem 1403]*
UpdateSegment	Metadata™ API *[Server APIs subsystem 1403]*
DeleteSegment	Metadata™ API *[Server APIs subsystem 1403]*
GetAllAnalysts	Manager API *[Manager APIs subsystem 1402]*

The Information Setup section 1515 is controlled by instances of the following classes: clnt_BuildRelationshipDlg, clnt_BuildMeasureDlg, clnt_BuildSegmentDlg and clnt_BuildRestrictDlg (all subclasses of the MFC's CDialog). If the user 1405 selects to modify a private segment or measure, the clnt_MeasureDefDlg and clnt_SegmentDefDlg objects will be responsible for traversing through the list of existing Analysts and InfoFrames™ and if the segment or measure is found, the objects will take the following actions:

In case of Delete:

- A message will be displayed to the User 1405 that deleting will cause some Analysts to no longer run correctly. The User 1405 will be presented with a list of Analysts that will be affected by this deletion. When an Analyst runs on a deleted segment or measure an error message will be returned.

In case of Modify:

- The newest segment/measure definition will always be used. The old definitions will be replaced.

The User change requests will be transferred to DAI (through the Server APIs subsystem) for immediate update of the Metadata™.

The Analyst Builder dialog box 1513 allows the User 1405 to select the parameters needed to generate a specific InfoFrame™ (see below). It also allows the User 1405 the option of Scheduling and/or defining Trigger conditions for an Analyst. To allow this to happen, the main Analyst dialog will prompt the User 1405 to complete a sequence of sub-dialogs: Measures, Segments, TimeSlice, Schedule, and Trigger.

Other portions of the User Interface subsystem 1401 (*i.e.*, Menus, Toolbar, or a Manager Window) invoke the Analyst Builder dialog 1513 either by passing it an existing Analyst object to view/edit, or by passing a NULL parameter, indicating that a new Analyst is to be created.

The clnt_AnalystSheet dialog will instantiate a clnt_InfoFrameRequest object when the User 1405 requests to "Save" on a new Analyst or "Save As" on an existing Analyst.

The clnt_AnalystSheet dialog makes the following requests to other subsystems:

Function Used	SubSystem
NewAnalyst	Manager API *[Manager subsystem 1402]*
UpdateAnalyst	Manager API *[Manager subsystem 1402]*
RunAnalystNow	InfoFrame™ Generation API *[Server APIs subsystem 1403]*
SetFrameDefinition	InfoFrameRequest API *[Manager subsystem 1402]*
GetFrameDefinition	InfoFrameRequest API *[Manager subsystem 1402]*

The clnt_AnalystSheet class instantiates sub-dialogs of the following classes: clnt_AnalystMeasurePage, clnt_AnalystSegmentPage, clnt_AnalystTimeSlicePage, clnt_AnalystSchedulePage, clnt_AnalystTriggerPage (all subclasses of MFC's CPropertyPage). These correspond to five panels within the main dialog box which the User 1405 will be led through in sequence.

Also, the Analyst subsystem 1513 will use clnt_MetaTree class and clnt_MeausreMap class which will provide access to the Metadate tables through Metadata™ API's.

The clnt_AnalystSheet subdialogs will be dynamically populated with the proper controls according to the User's 1405 selection of the Analyst type. The User input from the Dialog interfaces will be saved in a clnt_infoFrameRequest object and returned to Manager subsystem to be saved (and submitted for scheduling, if a Schedule is present -- see below regarding further details of the scheduler subsystem 18).

The clnt_AnalystMeasurePage makes the following requests to the other subsystems:

Function Used	SubSystem
GetName	InfoFrameDefinition API *[Manager subsystem 1402]*
SetName	InfoFrameDefinition API *[Manager subsystem 1402]*
GetFrameType	InfoFrameDefinition API *[Manager subsystem 1402]*
SetFrameType	InfoFrameDefinition API *[Manager subsystem 1402]*
GetTargetMeasure	InfoFrameDefinition API *[Manager subsystem 1402]*
SetTargtMeasure	InfoFrameDefinition API *[Manager subsystem 1402]*
GetComparisonMeasure	InfoFrameDefinition API *[Manager subsystem 1402]*
SetComparisonMeasure	InfoFrameDefinition API *[Manager subsystem 1402]*
GetAdditonalMList	InfoFrameDefinition API *[Manager subsystem 1402]*
SetAdditionalMList	InfoFrameDefinition API *[Manager subsystem 1402]*

The clnt_AnalystSegmentPage makes the following requests to other subsystems:

Function Used	SubSystem
GetTargetSegment	InfoFrameDefinition API *[Manager subsystem 1402]*
SetTargetSegment	InfoFrameDefinition API *[Manager subsystem 1402]*
GetComparisonSegment	InfoFrameDefinition API *[Manager subsystem 1402]*
SetComparisonSegment	InfoFrameDefinition API *[Manager subsystem 1402]*

GetAdditionalSList	InfoFrameDefinition API *[Manager subsystem 1402]*
SetAdditionalSList	InfoFrameDefinition API *[Manager subsystem 1402]*
GetPartitionList	InfoFrameDefinition API *[Manager subsystem 1402]*
SetPartitionList	InfoFrameDefinition API *[Manager subsystem 1402]*
GetParentPartition	InfoFrameDefinition API *[Manager subsystem 1402]*
GetParentPartition	InfoFrameDefinition API *[Manager subsystem 1402]*

The clnt_AnalystTimeSlicePage makes the following requests to other subsystems:

Function used	Subsystem
GetPeriodType	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetPeriodType	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetAnalysisType	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetAnalysisType	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetYearType	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetYearType	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetTrendInterval	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetTrendInterval	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetDuration	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetDuration	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetNumDuration	InfoFrameTimeSlice API *[Manager subsystem 1402]*

SetNumDuration	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetBasePeriod	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetBasePeriod	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetBaseThruPeriod	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetBaseThruPeriod	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetCompPeriod	InfoFrameTimeSlice API *[Manager subsystem 1402]*
SetCompPeriod	InfoFrameTimeSlice API *[Manager subsystem 1402]*
Operator=	InfoFrameTimeSlice API *[Manager subsystem 1402]*
GetTimeSlice	InfoFrameDefinition API *[Manager subsystem 1402]*

The clnt_AnalystSchedulePage makes the following requests from other subsystems:

Function	Subsystem
GetNumInterval	InfoFrameSchedule API *[Manager subsystem 1402]*
SetNumInterval	InfoFrameSchedule API *[Manager subsystem 1402]*
GetInterval	InfoFrameSchedule API *[Manager subsystem 1402]*
SetInterval	InfoFrameSchedule API *[Manager subsystem 1402]*
GetStartDate	InfoFrameSchedule API *[Manager subsystem 1402]*
SetStartDate	InfoFrameSchedule API *[Manager subsystem 1402]*
GetNumLimit	InfoFrameSchedule API *[Manager subsystem 1402]*
SetNumLimit	InfoFrameSchedule API *[Manager subsystem 1402]*

GetLimit	InfoFrameSchedule API *[Manager subsystem 1402]*
SetLimit	InfoFrameSchedule API *[Manager subsystem 1402]*
GetScheduleFlag	InfoFrameSchedule API *[Manager subsystem 1402]*
SetScheduleFlag	InfoFrameSchedule API *[Manager subsystem 1402]*
GetTriggerFlag	InfoFrameSchedule API *[Manager subsystem 1402]*
SetTriggerFlag	InfoFrameSchedule API *[Manager subsystem 1402]*
Operator=	InfoFrameSchedule API *[Manager subsystem 1402]*
SetSchedule	InfoFrameReuest API *[Manager subsystem 1402]*

The clnt_AnalystTriggerPage makes the following requests from other subsystems:

Function	Subsystem
GetTriggerList	InfoFrameTrigger API *[Manager subsystem 1402]*
SetTriggerList	InfoFrameTrigger API *[Manager subsystem 1402]*
GetMessageFlag	InfoFrameTrigger API *[Manager subsystem 1402]*
SetMessageFlag	InfoFrameTrigger API *[Manager subsystem 1402]*
GetFrameFlag	InfoFrameTrigger API *[Manager subsystem 1402]*
SetFrameFlag	Update the state of the frame generation action
GetAnalystList	InfoFrameTrigger API *[Manager subsystem 1402]*
SetAnlystList	InfoFrameTrigger API *[Manager subsystem 1402]*
Operator=	InfoFrameTrigger API *[Manager subsystem 1402]*
GetMeasure	Trigger API *[Manager subsystem 1402]*
SetMeasure	Trigger API *[Manager subsystem 1402]*
GetOperator	Trigger API *[Manager subsystem 1402]*
SetOperator	Trigger API *[Manager subsystem 1402]*
GetOperand1	Trigger API *[Manager subsystem 1402]*
GetOperand2	Trigger API *[Manager subsystem 1402]*
SetOperand1	Trigger API *[Manager subsystem 1402]*
SetOperand2	Trigger API *[Manager subsystem 1402]*
GetValue1	Trigger API *[Manager subsystem 1402]*

GetValue2	Trigger API *[Manager subsystem 1402]*
SetValue1	Trigger API *[Manager subsystem 1402]*
SetValue2	Trigger API *[Manager subsystem 1402]*
Operator=	Trigger API *[Manager subsystem 1402]*
SetTrigger	InfoFrameRequest API *[Manager subsystem 1402]*

The following is a list of user input requirements for each InfoFrame™ Type:

(R = Required, O = Optional)

clnt_MeasureDlg:

Analysis Type	Target Measure	Additional Measures	Comparison Measure
Change	R	O	
Segment Comparison	R	O	
Measure Comparison	R		R
Summarization	R	O	
Trend	R	O	

clnt_SegmentDlg:

Analysis Type	Target Segment	Additional Segments	Comparison Segment
Change	R	O	
Segment Comparison	R	O	R
Measure Comparison	R	O	
Summarization	R	O	
Trend	R	O	

clnt_TimeSliceDlg:

Analysis Type	Base Period	Comparison Period	Time Interval
Change	R	R	
Segment Comparison	R		
Measure Comparison	R		
Summarization	R		
Trend	R		R

The InfoFrame™ Viewer Window 1517 displays an InfoFrame™ on screen (see below). In addition to displaying the InfoFrame™ data, the Viewer 1517 supports the "Drill Down" capability by presenting *hot spots* to the User 1405, and generating the appropriate requests when a *hot spot* is selected. The InfoFrame™ Viewer also gives the User a capability to Annotate an InfoFrame™.

When InfoFrame™ Viewer 1517 is created, it receives the name of the InfoFrame™ file and a pointer to the InfoFrame™ object. This data is parsed (further processing is also done, including generating graphs from embedded data), then displayed.

The Parser capability within the InfoFrame™ Viewer module 1517 is also used for the SaveAs requests; the raw InfoFrame™ data is translated to standard HTML data (*i.e.*, MDT-specific graph data is translated into a graphical image in a standard format), and is written to a file in either ASCII or Unicode characters. The InfoFrame™ Viewer Window 1517 also supports the InfoFrame™ Print function. This functionality is built on the capabilities provided by the CDocument and CScrollView classes of MFC.

The InfoFrame™ Viewer subsystem 1517 makes the following requests to other subsystems.

Function Used	SubSystem
UpdateInfoFrame™	Manager API *[Manager subsystem 1402]*
DrillDown	Manager API *[Manager subsystem 1402]*

Parser: the clnt_Parser class provides the HTML parsing capability for the Client 12 through the following three functions:

Service Provided	Description
doParse	Called by the InfoFrame™ Viewer, this function parses the given HTML data, and returns a list of clnt_Tag objects, each representing an element of the HTML Data. The clnt_Tag objects can contain lists of sub-Tags, so that nesting is preserved.
SaveAsHTML_Unicode	Called by the Manager 1402 when the User 1405 requests to Save an HTML file. Parses the HTML data to replace any non-standard HTML elements with standard HTML data (for example, raw graph data must be transformed into a graphic image). Writes transformed data into a file, using Unicode characters.
SaveAsHTML_ASCII	Same as above, except characters are written out as ASCII.

Viewer: the Viewer is implemented using the MFC Document/View architecture. Class clnt_Viewer is a subclass of CScrollView (MFC), which provides the automatic scrolling. Class clnt_ParserDoc is a subclass of CDocument. On creation, it instantiates a clnt_Parser object to parse the HTML Data. The clnt_Viewer then traverses the returned list of clnt_Tag objects and places their visual representations in the Window.

The following collection of controls are used by the user interface subsystem 1401:

clnt_TreeCtrl. All dialog controls which will be representing segments and/or partitions inherit from this class rather than from the MFC's CTreeCtrl. clnt_MetaTree control also inherits from this class.

clnt_MetaTree. This control is used to represent the Metadata™ segments and partitions in a hirerchical format. The following dialogs subclass this control: clnt_AnalystSegmentPage, clnt_BuildSegmentDlg, clnt_RestrictMeasureDlg.

clnt_TopLevelSegmentCombo. This conttrol is used to represent all Metadata™ top level segments in a DropDown ComboBox. The following dialogs subclass this control: clnt_AnalystSegmentPage, clnt_BuildSegmentDlg, clnt_RestrictMeasureDlg.

clnt_DurationCombo. This control represents the user's conditional operator choices in a dropdown combobox format. The following dialogs subclass this control: clnt_AnalystPeriodPage, clnt_AnalystSchedulePage.

clnt_OperatorCombo. This control represents the user's conditional operator choices in a dropdown combobox format. The following dialogs subclass this control: clnt_AnalystPeriodPage, clnt_AnalystSchedulePage.

clnt_DateEdit. This control is used to represent the locale date. It validates the user entry and formats the date properly for the locale. The following dialogs subclass this control: clnt_AnalystPeriodPage, clnt_AnalystSchedulePage.

lnt_ReadOnlyListBox. This control is used for a non-select listbox. There is no dependencies from other subsystems. The following dialog subclasses this control: clnt_BuildSegmentDlg

The clnt_MetaTree control uses the following services from other subsystems:

Function Used	SubSystem
GetSegment	Metadata™ API *[Server APIs subsystem 1403]*
GetPartition	Metadata™ API *[Server APIs subsystem 1403]*

The clnt_TopLevelSegmentCombo uses the following services from other subsystems:

Function Used	SubSystem
GetSegment	Metadata™ API *[Server APIs subsystem 1403]*

The clnt_MeasureCombo uses the following services from other subsystems:

Function Used	SubSystem
GetBasicMeasure	Metadata™ API *[Server APIs subsystem 1403]*
GetCompositeMeasure	Metadata™ API *[Server APIs subsystem 1403]*

The Administrator Interface 1516 consists of two tasks: User Account setup, and Metadata™ Builder. The User Accounts setup dialog allows the MDTA (Administrator) to create and manager User accounts, including login name, password, and User type. The Metadata™ Builder allows the MDTA to define Dimensions, Attributes, and Basic Measures, to create Segments, map columns for Time values, and define Year types.

The User Accounts screen utilizes the clnt_UserLogin class from the Server APIs subsystem 1403. The Metadata™ Builder screens utilize nearly all Metadata™ functions provided by the Server APIs subsystem 1403. This includes the services of classes clnt_Communications, clnt_Dimension, clnt_Attribute, clnt_BasicMeasure, and

clnt_Segment. It also uses the clnt_Schema class for access to the Data Warehouse schema.

The User Accounts dialog is controlled by an instance of clnt_UserAccountsDlg, a subclass of MFC's CDialog. The interface that clnt_UserAccountsDlg presents to the rest of the system is the standard for CDialog objects; the instance is constructed, and then DoModal() is called to display the dialog. The call to DoModal() returns only when the User 1405 presses the "Cancel" or "Close" button.

The Metadata™ Builder dialog may be a "wizard" style dialog, meaning that it presents a series of sub-dialogs in a pre-determined order. The User 1405 may press the "Next" and "Back" buttons to traverse the list of sub-dialogs, and may press "Cancel" to exit from the Metadata™ Builder. The "frame" of the wizard is implemented by class clnt_MasterSetup, which is a subclass of MFC's CPropertySheet. The constructor of clnt_MasterSetup creates one instance each of the dialog "pages" (clnt_AttributeDefinition, clnt_AttributeMapping, clnt_AttributeValueDefinition, clnt_AutomaticSegments, clnt_BasicMeasureDefinition, clnt_BasicMeasureMap, clnt_DimensionDefinition, clnt_Joins, clnt_TimeDimension, clnt_YearDefinition). The pages are loaded into the "wizard" automatically when it is displayed. This is transparent to the rest of the Client application 12, which simply constructs the Metadata™ Builder and calls DoModal() on the instance.

Each of the "pages" loads its initial display data through calls to the ServerAPIs 1403 Metadata™ classes, and each page responds to the "Save" button by updating its data through the ServerAPIs 1403.

The clnt_MasterSetup has one linked list for each type of Metadata™ used. Each list contains zero or more clnt_SetupObject objects. The clnt_SetupObject object contains two data members: one pointer to a CObject and one clnt_ObjectState enumeration. clnt_ObjectState can take on one of four values: STATE_EXISTING, STATE_NEW, STATE_DELETED, STATE_MODIFIED. These linked lists are available

to every "wizard" page. Every time the user 1405 adds, deletes or modifies a Metadata™ object, it is added to the appropriate linked list. These linked lists are used to determine which objects to display to the user 1405 and which ones to hide from the user 1405. The linked lists are also used by the "CANCEL" and "SAVE" buttons. When the user 1405 presses the "CANCEL" button, all objects in the linked lists are deleted. When the user 1405 presses the "SAVE" button, all objects in the linked lists are accessed. If the value of the enumeration is STATE_EXISTING the object is deleted from the list. If the value is STATE_NEW the object is added to the Metadata™ on the server and deleted from the list. If the value is STATE_DELETED the object is deleted from the Metadata™ on the server and the object is deleted from the list. If the value is STAT_MODIFIED the object is updated in the Metadata™ on the server and the object is deleted from the list.

The "SAVE" button on the "wizard" page adds, deletes and modifies objects in a certain order. For deleting objects, the following table lists the object to be deleted in the left column and the associated objects in the right column that will be deleted from the linked lists on the Client 12 if they exist. The row order in the left column defines which object will be deleted, added or modified first. Dimensions would be added first and Year Definitions would be added last.

Object	Associated objects
Dimension	Attribute, Segment
Attribute	Enumerate Attribute Value, Restricted Integer Attribute, Restricted Float Attribute, Segment, Attribute Measure Join, Attribute Attribute Join
Enumerated Attribute Value	<none>
Restricted Integer Attribute	<none>
Restricted Float Attribute	<none>
Segment	Numerical Attribute Restriction
Numerical Attribute Restriction	<none>
Enumerate String Attribute Restriction	<none>

Partition	<none>
Basic Measure	Attribute Measure Join
Composite Measure	Constant, Segment List, Attribute Measure Join
Constant	<none>
Segment List	<none>
Attribute Measure Join	<none>
Attribute Attribute Join	<none>
Measure Relationship	Measure Relation Range Restriction, Measure Relation Magnitude Restriction, Measure Relation Segment Constraint
Measure Relation Range Restriction	<none>
Measure Relation Magnitude	<none>
Measure Relation Segment Constraint	<none>
Time Definition	<none>
Time Mapping	<none>
Year Definition	<none>

When the user deletes an object that already exist in the Metadata™ 25 on the server 34, just that object is deleted. The "associated objects" for that object will be deleted by the DAI subsystem 14.

The Manager Windows 1512 give the User 1405 access to all types of data which are stored by the Manager subsystem 1402: Folders, Analysts, and InfoFrames™, as well as information about Pending InfoFrames™.

There are four types of Manager Windows 1512, each offering a different view of this data:

- Analyst list (flat list of all Analysts)
- InfoFrame™ list (flat list of all InfoFrames™)
- Folder View (includes Folder hierarchy; shows InfoFrames™ & Analysts in current Folder)
- Pending Queue (flat list of InfoFrames™ pending in the DAI 14).

Note that the Pending Queue window is included with the other three Manager Windows because of its similarity in construction and interface behavior; the data it displays is actually quite distinct from that of the other three Windows.

Drag-and-Drop features are also supported by the Manager Windows 1512. The Analyst list, InfoFrame™ list, and Folder View can be the source of a "drag" operation (Users may drag one Folder, one or more Analysts, or one or more InfoFrames™). The Folder View may also be the destination of a "drag" operation.

The first three Manager Window types (Analyst list, InfoFrame™ list, Folder view) use the following services from other subsystems:

Function Used	SubSystem
GetRootFolder	Manager API *[Manager subsystem 1402]*
GetTrashBin	Manager API *[Manager subsystem 1402]*
GetAllAnalysts	Manager API *[Manager subsystem 1402]*
GetAllInfoFrames™	Manager API *[Manager subsystem 1402]*
NewFolder	Manager API *[Manager subsystem 1402]*
RemoveFolder	Manager API *[Manager subsystem 1402]*
MoveFolder	Manager API *[Manager subsystem 1402]*
SetFolderName	Manager API *[Manager subsystem 1402]*
MoveAnalyst	Manager API *[Manager subsystem 1402]*
RemoveAnalyst	Manager API *[Manager subsystem 1402]*
MoveInfoFrame™	Manager API *[Manager subsystem 1402]*
RemoveInfoFrame™	Manager API *[Manager subsystem 1402]*
EmptyTrash	Manager API *[Manager subsystem 1402]*
GetChildFolders	Folder API *[Manager subsystem 1402]*
GetInfoFrames™	Folder API *[Manager subsystem 1402]*
GetAnalysts	Folder API *[Manager subsystem 1402]*
RemoveFolder	Folder API *[Manager subsystem 1402]*
RunAnalystNow	InfoFrame™ Generation API *[Server API subsystem 1403]*
ViewInfoFrame™	InfoFrame™ Viewer Window *[User Interface subsystem 1401]*
run AnalystBuilder dialog	Analyst Builder *[User Interface subsystem 1401]*

The fourth Manager Window, Pending Queue, uses the following services from other subsystems:

Function Used	SubSystem
GetStatus	InfoFrame™ Generation API *[Server API subsystem 1403]*
CancelAnalyst	InfoFrame™ Generation API *[Server API subsystem 1403]*

Each of the four Manager Windows 1512 is controlled by a *frame* object and one or more *control* objects placed within the *frame*. In all four cases, the *frame* is represented by just one class, clnt_ManagerWnd, a subclass of CMDIChildWnd from MFC. The clnt_ManagerWnd object is parameterized on instantiation to indicate which *control* object(s) it should construct. As the superclass would suggest, it behaves as a standard MDI Child Window.

The *control* objects within the frame window inherit from MFC classes which are, in turn, wrappers for standard MS-Windows Controls. Classes clnt_AnalystCtrl, clnt_InfoFrameCtrl, and clnt_PendingCtrl each inherit from CListCtrl, and display their data in "columned" lists. Class clnt_FolderCtrl inherits from CTreeCtrl to display the tree-like hierarchy of the MDT Folders. These classes are instantiated, as needed, by the clnt_ManagerWnd, depending on the "style" flag it receives: clnt_AnalystCtrl is used in ANALYSTS mode and FOLDERS mode; clnt_InfoFrameCtrl is used in INFOFRAMES and FOLDERS modes; clnt_FolderCtrl is used in FOLDERS mode and by the clnt_SaveAs dialog box (a part of the Analyst Builder); clnt_PendingCtrl is used in PENDING mode.

When the User 1405 begins a drag-and-drop operation, the source window of the drag constructs an instance of clnt_DragWnd, which then controls the remainder of the drag-and-drop. The clnt_DragWnd is given a pointer to the object or list of objects being dragged, and also an indication of the type of object being dragged. It then sends a message to any window the cursor passes over, asking whether it is "OK" to drop the

object in that window. The windows which support drops are clnt_FolderCtrl and clnt_Toolbar (see section 3.2.3). When the User 1405 releases the mouse 21 button, the clnt_DragWnd sends a message to the destination window requesting it to accept the dropped item(s), and also sends a message to the source window indicating that the drop was completed.

The Manager subsystem 1402 handles all functions related to manipulating, storing, and retrieving Folder 100 hierarchies, and the InfoFrames™ and Analysts that are stored in those Folders. Because all functions related to storing and retrieving this data are encapsulated in the Manager subsystem 1402, there will be minimal impact on the other Client subsystems if the Folders/InfoFrames™/Analysts data store moves onto the Server tier 32 in an alternate embodiment of the present invention.

As can be seen Fig. 15, the Manager 1402 provides four APIs: Manager 1521, Folder 1522, Analyst 1523, and InfoFrame™ 1524. These APIs correspond to four classes which are described in the following sections. The main class in the Manager subsystem 1521 is the clnt_Manager class. Three data object classes: clnt_Folder, clnt_InfoFrameRequest, and clnt_InfoFrame™, are used by the clnt_Manager, and by other subsystems. Access to Manager functions normally begins with a call to the clnt_Manager itself, requesting a list of Folders, Analysts, or InfoFrames™. The objects which are returned by these queries can then be displayed to the User 1405 for viewing and/or manipulating. Requests for changes to any of the data objects pass through the clnt_Manager, which handles storing the changes on disk and, as applicable, sending the changes to the Server API subsystem 1403.

The Manager subsystem 1402 also provides a "TrashBin" capability; that is, when a request to delete an Analyst or an InfoFrame™ is received, the object is placed in the TrashBin, and not actually deleted until the next EmptyTrash command is received. The TrashBin is persistent between sessions of the Client 12. The TrashBin is implemented as an instance of the clnt_Folder class.

There is exactly one instance 1521 of the clnt_Manager class in the Client application 12. In order to ensure that only one instance will be created, and that it will be safely globally available, the class uses the "Singleton" design pattern (as described in Gamma, et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995, ISBN 0-201-63361-2). In this pattern, the class provides a static member function which returns a pointer to the one instance of itself. The function automatically creates the instance the first time it is called. The constructor of the class is made protected, thus ensuring that the class is never instantiated elsewhere.

The clnt_Manager class handles the following requests from other subsystems:

Service Provided	Description
GetManager	Static class function which returns a pointer to the one instance of clnt_Manager. See above.
GetRootFolder	Returns pointer to top clnt_Folder object.
GetTrashBin	Returns pointer to TrashBin (actually a clnt_Folder object).
GetAllAnalysts	Returns a list of all Analysts, without regard to Folder hierarchy.
GetAllInfoFrames™	Returns a list of all InfoFrames™, without regard to Folder hierarchy.
NewFolder	Creates a new Folder; parameter indicates the parent for the new Folder; returns pointer to the newly-created clnt_Folder object.
RemoveFolder	Removes the given clnt_Folder object; all of its sub-folders, Analysts, and InfoFrames™ are also removed.
MoveFolder	Moves the given Folder to a new Parent Folder.
NewAnalyst	Stores a new clnt_InfoFrameRequest object in the given Folder, sends to ScheduleAnalyst Server API, if a Schedule is present.
UpdateAnalyst	Stores changes to an existing clnt_InfoFrameRequest object, sends to UpdateAnalyst Server API, if a Schedule is present.

MoveAnalyst	Moves clnt_InfoFrameRequest object to a different Folder.
RemoveAnalyst	Deletes the given clnt_InfoFrameRequest object; if a Schedule is present, sends a CancelAnalyst to the Server API subsystem.
UpdateInfoFrame™	Stores changes to an existing clnt_InfoFrame™ object (normally changes to its HTML data, when annotations are added or raw data is processed into a graph, for example).
MoveInfoFrame™	Moves the given clnt_InfoFrame™ object to a different Folder.
RemoveInfoFrame™	Deletes the given clnt_InfoFrame™ object.
DrillDown	If requested DrillDown Frame is already generated, returns that Frame; if not, sends the Frame Generation request to Server API.
SaveInfoFrameAsMDTFile	Creates a file that can be e-mailed, etc. File is an "MDT InfoFrame™ File"—only useable by someone who has Client software 12.
SaveInfoFrameAsHTMLFile	Creates a file that can be e-mailed, etc. File is a standard HTML 3.0 file, viewable by any HTML Browser program. A parameter to the function indicates if ASCII or UNICODE output was requested by the User.
ImportMDTFile	Reads in a file previously created by SaveInfoFrameAsMDTFile command, stores it as an InfoFrame™ object in a Folder. The InfoFrame™ is then available for viewing through standard mechanisms.
EmptyTrash	Completely deletes all items currently in the TrashBin.

The clnt_Manager object uses the following services from other subsystems:

Function Used	SubSystem
ScheduleAnalyst	Analyst API *[Server API subsystem 1403]*
UpdateAnalyst	Analyst API *[Server API subsystem 1403]*
CancelAnalyst	Analyst API *[Server API subsystem 1403]*
GetStatus	InfoFrame™ Generation API *[Server API subsystem 1403]*
RetrieveFrame	InfoFrame™ Generation API *[Server API subsystem 1403]*

Instances 1522 of class clnt_Folder are instantiated and deleted only by the clnt_Manager object. Other subsystems gain access to clnt_Folder instances starting with the clnt_Manager's GetRootFolder() or GetTrashBin() functions.

The clnt_Folder object handles the following requests from other subsystems:

Service Provided	Description
GetFolderName	Returns name of this Folder.
SetFolderName	Changes the name of this Folder.
GetChildFolders	Returns a list of clnt_Folder objects which are "children" of this Folder.
GetInfoFrames™	Returns a list of clnt_InfoFrame™ objects which are stored in this Folder.
GetAnalysts	Returns a list of clnt_InfoFrameRequest objects which are stored in this Folder.
RemoveFolder	Removes the given clnt_Folder object; all of its sub-folders, Analysts, and InfoFrames™ are also removed.

Instances 1523 of class clnt_InfoFrameRequest are created by the clnt_Manager object (when restoring saved Analysts from disk) and by the clnt_AnalystBuilder dialog class (when creating a new Analyst or doing a SaveAs on a

current Analyst). Other subsystems normally access Analyst objects by retrieving them from their Folder (clnt_Folder::GetAnalysts()). Instances of clnt_InfoFrameRequest are only deleted by the clnt_Manager object.

The clnt_infoFrameRequest class handles the following requests from other subsystems:

Service Provided	Description
GetName	Returns name of this Analyst.
SetName	Assigns a new name for this Analyst.
GetRequestID	Return a unique request ID assigned by Manager
SetRequestID	Assigns a unique request ID to the Analyst Request
GetUserName	Returns the user name
SetUserName	Assigns a user name to the Analyst Request
GetFrameDef	Returns the clnt_InfoFrameDefinition object for this Analyst.
SetFrameDef	Updates the Analyst's FrameDefinition object.
GetSchedule	Returns the clnt_Schedule object for this Analyst.
SetSchedule	Updates the Analyst's Schedule object.
GetTrigger	Returns the clnt_Trigger object for this Analyst.
SetTrigger	Updates the Analyst's Trigger object.
GetContainingFolder	Returns the pointer to the containig foldr object
SetContainingFolder	Updates the pointer to the containing folder object

The clnt_InfoFrameRequest class does most of its work through three helper classes. The clnt_InfoFrameDefinition class stores a description of the InfoFrame™ Generation request that will be sent when this Analyst is run (or scheduled).

The clnt_infoFrameDefinition class handles the following requests from other subsystems

Service Provided	Description
GetFolderID	Return the Folder ID assign to the analyst by clnt_Folder object
SetFolderID	Assigns the Folder ID to the Analyst.

GetAnalysisType	Returns the type of analysis Selected for this request.
SetAnalysisType	Updates the type of analysis selected for this request.
GetTargetMeasure	Returns the target measure selected for this analysis. Required
SetTargetMeasure	Updates the target measure selected for this analysis. Required.
GetComparisonMeasure	Returns the comparison Measure selected for this analysis. Required only for Measure Compariosn Analysis.
SetComparisonMeasure	Updates the Compariosn Meausre Selected for this analyisis. Required only for Measure Comparison Analysis
GetAdditionalMList	Returns a list of Additional measure Objects selected for this analysis. Optional.
SetAdditionalMList	Updates the List of Additional measure Objects slected for this analysis. Optional
GetTargetSegment	Returns the target segment selected for this analysis. Required
SetTargetSegment	Updates the target Segment selected for this analysis. Required.
GetComparisonSegment	Returns the comparison Segment selected for this analysis. Required only for Segment Compariosn Analysis.
SetComparisonSegment	Updates the Compariosn Segment Selected for this analyisis. Required only for Segment Comparison Analysis
GetAdditionalSList	Returns a list of Additional measure Objects selected for this analysis. Optional.
SetAdditionalSList	Updates the List of Additional Segment Objects slected for this analysis. Optional
GetPartitionList	Returns the List of selected target Partitions. Optional.
SetPartitionList	Updates the List of selected target Partitions. Optional.
GetParentPartition	Returns the target segment's parent partition. Required only if target segment is not top level segment.

SetParentPartition	Updates the target segment's parent partition. Required only if target segmetn is not top level segment.
GetTimeSlice	Returns the pointer to the timeslice object for this analysis Required.
SetTimeSlice	Updates the pointer to the timeslice object for this analyusis. Required.
Operator=	Copies the object into another

The clnt_InfoFrameTimeSlice class handles the following requests from other subsystems:

Service Provided	Description
GetPeriodType	Returns the type of timeslice selected for the request.
SetPeriodType	Updates the type of timeslice selected for the request.
GetAnalysisType	Returns the type of analysis selected for the request.
SetAnalysisType	Updates the type of analysis selected for the request.
GetYearType	Returns the type of year definition selected for the request
SetYearType	Updates the type of year definition selected for the request
GetTrendInterval	Returns the interval duration. Required only for Trend Analysis.
SetTrendInterval	Updates the interval duration. Reuired only for Trend Analysis.
GetDuration	Returns the time duration.
SetDuration	Updates the time duration.
GetNumDuration	Returns the number of durations.
SetNumDuration	Updates the number of durations.
GetBasePeriod	Returns the Specific Date's base period.
SetBasePeriod	Updates the Specific Date's base period.
GetBaseThruPeriod	Returns the Specific Date's thru period.
SetBaseThruPeriod	Updates the Specific Date's thru period.
GetCompPeriod	Returns the Specific Date's Comparison Period. Required only by Change Analysis.
SetCompPeriod	Updates the Specific Date's Comparison Period. Required only by Change Analysis.
Operator=	Copies one TimeSlice object inot another.

The clnt_InfoFrameSchedule class stores definition of a schedule for the Analyst.

The clnt_infoFrameSchedule class handles the following requests from other subsystems:

Service Provided	Description
GetNumInterval	Return Number of intervals the report should run.
SetNumInterval	Update the number of intervals the report should run.
GetInterval	Return the duration for the interval the report should run.
SetInterval	Updates the duration for the interval the report should run.
GetStartDate	Rutun the date to which the report is scheduled to start runing.
SetStartDate	Updates the date to which the report is scheduled to start running.
GetNumLimit	Retuns the number of time periods the reports is scheduled to run.
SetNumLimit	Updates the number of time periods the report os scheduled to run,
GetLimit	Return the duration for the number of times the report is scheduled to run.
SetLimit	Updates the duration for the number of times the report is scheduled to run.
GetScheduleFlag	Returns the enabling or disabling state of the schedule.
SetScheduleFlag	Updates the enabling or disabling state of the schedule.
GetTriggerFlag	Returns the enabling or disabling state of the trigger definition.
SetTriggerFlag	Updates the enabling or disabling state of the trigger definition.
Operator=	Copies one Schedule object into another

The clnt_InfoFrameTrigger class handles definition of trigger conditions to be checked before the Analyst is run. The clnt_infoFrameTrigger class handles the following requests from other subsystems:

Service Provided	Description
GetTriggerList	Return a list of triggers defined by the analyst
SetTriggerList	Updates a list of triggers defined by the analyst
GetMessageFlag	Return the enable/disable state depending on user selection of the action to be taken. In this case a message will be generated if trigger becomes true.
SetMessageFlag	Update the state of the message generation action.
GetFrameFlag	Return the enable/disable state depending on user selection of the action to be taken. In this case a Frame will be genereated if trigger becomes true.
SetFrameFlag	Update the state of the frame generation action
GetAnalystList	Return a list of analysts to be generated if the trigger becomes true. If List empty, this action is not selected.
SetAnlystList	Update the list of analyst to be generated if the trigger becomes true. If List empty , this action is not selected
Operator=	Copies the object into another

The class clnt_trigger contains a single trigger condition. A list of clnt_trigger objects will be ANDed and defined as a single trigger. The clnt_Trigger class handles the following requests from other subsystems:

Service Provided	Description
GetMeasure	Returns the measure selected.
SetMeasure	Updates the measure seleceted
GetOperator	Returns the operator selected
SetOperator	Updates the operator selected
GetOperand1	Returns the first operand measure
GetOperand2	Returns the second operand measure if operator is Between or not Between.

SetOperand1	Updates the first operand measure
SetOperand2	Updates the second operand measure if operator is Between or not Between.
GetValue1	Returns the first operand value
GetValue2	Returns the second operand value
SetValue1	Update the first operand value
SetValue2	Updates the second operand value, if Operator is Between or not Between.
Operator=	Copies the object into another

Each instance 1523 of clnt_InfoFrameRequest must have a clnt_infoFrameDefinition object; the clnt_InfoFrameSchedule and clnt_Trigger objects are optional.

Instances 1524 of class clnt_InfoFrame™ are instantiated by the clnt_Manager object (when restoring saved Analysts from disk or receiving a new Frame from the Server API). Other subsystems normally access InfoFrame™ objects by retrieving them from their Folder (clnt_Folder::GetInfoFrames™ ()). Instances of clnt_InfoFrame™ are only deleted by the clnt_Manager object.

The clnt_InfoFrame™ class handles the following requests from other subsystems:

Service Provided	Description
GetName	Returns name of this InfoFrame™.
GetHTMLFile	Returns name of file containing HTML Frame data is stored.
UpdateHTMLFile	Informs InfoFrame™ that it's data file has been updated (may be required for annotation, drill-down, graph generation).

The Server API subsystem 1403 encapsulates all functions which require communication with the MDT Server 32 (DAI 14). This isolates the User Interface 1401 and Folder Manager 1402 subsystems from specific knowledge about the Client-Server interface, keeping them independent of minor changes, etc.

As seen in Fig. 15, the Server APIs 1403 can be divided into four API modules: Metadata™ 1531, InfoFrame™ Generation 1532, Data Warehouse 1533, and Session Management 1534. The Server APIs subsystem 1403 also includes a set of internal routines for talking to the CSM 1404, which are shared by the four APIs. Each module is described below.

The Metadata™ API 1531 handles all Client 12 requests to view or modify portions of the MDT Metadata™ 25. Again the Metadata™ 25 resides on the server 34, and the Client 12 retrieves pieces of the Metadata™ 25 as needed, via the DAI 14.

The Metadata™ API 1531 provides the following services to other Client subsystems:

Service Provided	Description
GetDimensions	Returns a list of clnt_Dimension objects representing all Dimensions.
AddDimension	Add a new Dimension.
UpdateDimension	Update an existing Dimension.
DeleteDimension	Remove an existing Dimension.
GetDimensionPartitions	Returns a list of clnt_Partition objects for all Child Partitions within a given Dimension.
GetPartitions	Returns a list of clnt_Partition objects for all Child Partitions within a given Segment.
AddPartition	Add a new Partition.
UpdatePartition	Update an existing Partition.
DeletePartition	Remove an existing Partition.
GetSegments	Returns a list of clnt_Segment objects for all defined Segments within a given Partition.
AddSegment	Add a new Segment.
UpdateSegment	Update an existing Segment.
DeleteSegment	Remove an existing Segment.
GetMeasures	Returns 3 lists: Basic Measures, Composite Measures, and all Measures which are accessible to the User (some Measures are used internally in the Client code).
AddMeasure	Add a new Measure.
UpdateMeasure	Modify an existing Measure.

DeleteMeasure	Remove an existing Measure.
GetMeasureRelationship	Retrieve a Measure Relationship.
AddMeasureRelationship	Add a new Measure Relationship.
UpdateMeasureRelationship	Update an existing Measure Relationship.
DeleteMeasureRelationship	Remove an existing Measure Relationship.
GetRelationships	Retrieve possible Relationship types for a given Attribute.
GetRange	Retrieve range of values for a given Attribute.

The Metadata™ API uses the Communications Services module (see below) to communicate with the CSM 1404.

Several classes work together to provide the set of services listed in the table above. Class clnt_Dimension has public methods: GetDimensions (a static class method), AddDimension, UpdateDimension, DeleteDimension, and GetDimensionPartitions, AddDimensionPartition, DeleteDimensionPartition. Class clnt_Partition has public methods: GetSegments, AddSegment, DeleteSegment, UpdatePartition. Class clnt_Segment has public methods: GetPartitions, AddPartition, DeletePartition, UpdateSegment. Measure functions are represented by two classes: clnt_BasicMeasure and clnt_CompositeMeasure.

The InfoFrame™ Generation API 1532 contains all functions related to requesting that the DAI 14 run or schedule an Analyst, and retrieving status and completed InfoFrames™ from the DAI 14. Functions in this API 1532 are used by the Manager subsystem 1402 and the User Interface subsystem 1401.

The InfoFrame™ Generation API 1532 provides the following services to other Client subsystems:

Service Provided	Description
GetInfoFrameGenerationInstance	Returns a pointer to the one-and-only-one instance of clnt_InfoFrameGeneration.
GetStatus	Query the DAI for list of currently pending and/or completed InfoFrames™.

RetrieveFrame	Retrieve a specific completed InfoFrame™ from the DAI.
RunAnalystNow	Send an InfoFrame™ Generation request to the DAI for immediate processing.
ScheduleAnalyst	Send an InfoFrame™ Generation request to the DAI with a schedule on which to run it.
UpdateAnalyst	Send the DAI a modification to an existing, scheduled Analyst.
CancelAnalyst	Cancel a previously scheduled InfoFrame™ Generation request.

The InfoFrames™ API 1532 also uses the Communications Services module to communicate with the CSM 1404.

The InfoFrame™ Generation API 1532 functions listed above are public members of the clnt_InfoFrameGeneration class. The clnt_InfoFrameGeneration class will be instantiated only once, using the "Singleton" pattern (described previously).

The Data Warehouse API 1533 provides services related to setting up Metadata™ 25, at which time the MDTA (Administrator) needs access to information about the schema of the Data Warehouse 24. This API 1533 cis encapsulated in the clnt_Schema class. The Data Warehouse API 1533 provides the following services to other MDT Client subsystems:

Service Provided	Description
GetTables	Returns a list of names of Tables from the current Schema data.
GetColumns	Returns a list of names of Columns from a given Table.
GetPrimaryKeys	Returns a list of names of Primary Keys in a given Table.
GetForeignKeys	Returns a list of Foreign Keys in a given Table.
ForeignToPrimary	Returns a list of Primary Keys associated with a given Foreign Key in a given Table.
PrimaryToForeign	Returns a list of Foreign Keys associated with a given Primary Key in a given Table.

LoadSchema	Loads schema from server; returns True if successful.

The Data Warehouse API 1533 also uses the Communications Services module to communicate with the CSM 1404.

The Data Warehouse API 1533 is encapsulated in the clnt_Schema class. The clnt_Schema class has public member functions which correspond directly to the API calls described in the table above. The LoadSchema function loads all of the Data Warehouse schema onto the Client for the other API functions to access. The schema is discarded after each use.

The Session Management API 1534 contains functions related to establishing a session with the MDT Server (and to closing the connection when exiting). This includes functions related to User and Password management. The Session Management API provides the following services to other MDT Client subsystems:

Service Provided	Description
GetSessionManager	Returns pointer to the one-and-only-one instance of class clnt_SessionManager.
Connect	Establish a connection to the Server and attempt to authenticate the given User Name and Password.
Disconnect	Orderly shutdown of our connection to the Server.
UpdateUserPassword	Change a User's password on the Server.

The Session Management API 1534 also uses the Communications Services module to communicate with the CSM 1404.

The Session Management functions listed above are public member functions of class clnt_SessionManager. Class clnt_SessionManager is instantiated once and only once, using the "Singleton" pattern.

The Communication service encapsulates functions for talking to the DAI 14 via the CSM 1404. These functions are shared by the four APIs 1531, 1532, 1533 and

1534 within the Server APIs subsystem 1403. The Communications Services capabilities are encapsulated in class clnt_Communications, which will be a private superclass for all other classes in the Server APIs subsystem 1403.

3. Data Abstraction Intelligence (DAI) Subsystem 14

The data abstraction intelligence subsystem 14 is described in further detail below.

A key feature of the present invention (also referred to as Management Discovery Tool™ or MDT) is that it allows the user 1405 to easily choose different levels of granularity in the viewing and understanding of the objects of interest, and have these different levels reflected in InfoFrames™ generated by the MDT. For example, when the user 1405 thinks of the concept of product, he or she may mean all products marketed by the Enterprise, or, more likely, some interesting subset of all products. This subset can be defined by restrictions on the attributes of product, either a single restriction such as men's clothing (defined as the product Department = men's clothing), or multiple restrictions, such as expensive men's suits made by Designer Y (similarly defined by restrictions on Department, Price, and Manufacturer).

One of the insights in the design of the present invention is the notion that such subsets form hierarchies, and that these hierarchies may provide a convenient and powerful way for the user to both think about their and select relevant levels or granularities for the production of InfoFrames™. An important technical point, but one that is kept partially hidden from the user 1405, is how related segments form partitions.

The Management Discovery Tool™ of the present invention sits on top of a data warehouse 24, a single logical, consistent view of an enterprise's data. Typically, there are many different ways to store data; that is, there are different table structures or schema for a given set of data. For most, if not all, enterprises, there exists a small set of fundamental data types that are the lowest level of granularity and correspond to, typically, specific entities like product, customer, transaction, and the like. These entities can be

thought of as having a set of attributes associated with them, and these attributes can have values. In the relational model, this corresponds to a table of entities, with the attributes mapping into columns. Again, physically, they may be stored as several tables, but conceptually, this is what MDT is working with.

Fig. 16 illustrates the hierarchy formed by the user 1405 choosing the Department attribute 1601, then selecting segment Men's Clothing Department 1602 of the attribute; of choosing the Product-type attribute 1603, then selecting the segment Shirts 1604 of that attribute; the Manufacturer Attribute 1605, then Designer X 1606; and finally, the Size attribute 1607, and a particular partition 1608/1609. The final Segment of interest is: "Designer X Men's Shirts". Note that this segment could have been reached in several different ways, in particular, by any order of relevant attributes.

This scheme creates several requirements for other parts of MDT. First, the attributes of each dimension must be available. Second, the legitimate values for each attribute must also be made available. For finite domains, these must be listed for the user 1405; for infinite domains, current minimum and maximum values may be useful. There is a subtly here that can perhaps be avoided at first: the possible attribute values may not, in fact, be appropriate in all cases. In the above example, all Manufacturers may not make Men's Shirts. It may be possible to query the database for legitimate values, or store this information as additional Metadata™.

An important requirement of the present invention is to provide the ability to save and re-use user-defined segments. If "Designer X Men's Shirts" is an important category to the user 1405, he or she should be able to save that category and re-use it in the generation of InfoFrames™. Our approach allows users to define segments and automatically keeps these segments in a hierarchy. The table below provides a set of named segments and their definition, i.e. the restrictions on their attributes. All segments are on the dimension "Product".

Name	Dept.	Maker	Type	Size
Men's Clothing	MC			
Men's Shirts	MC		Shirts	
Designer X Shirts		Designer X	Shirts	
Designer X Men's Shirts	MC	Designer X	Shirts	
Men's Pants	MC		Pants	
Large Men's Shirts	MC		Shirts	Large
Large Men's Pants	MC		Pants	Large
Gap Products		Gap		
Designer X Products		Designer X		
Guess Products		Guess		
Medium and Large Men's Clothing	MC			Medium+ large
Small Men's Clothing	MC			Small
Small Men's Shirts	MC	Shirts		Small

The segments described above give rise to the segment hierarchy depicted in Fig. 17. Note that there is an inherent ambiguity in this structure: a given Segment can belong to several different partitions and have children beneath it whom are in several different partitions. For example, the Segment "Designer X-men's-shirts" belongs to two partitions. The first partition is on "Designer X-shirts" and uses the value "men's" to further discriminate (note the "Other" segment would now mean "Designer X-shirts-but-not-for-men". This segment is also part of a partion on "men's-shirts", and restricts the Manufacturer attribute to be Designer X. Here, the "Other" segment indicates "men's-shirts-made-by-everybody-but-Designer X".

In order to resolve the ambiguity, the notion of partition must be made explicit. Thus, Fig. 17 may be re-drawn as shown in Fig. 18 to include partitions (dark boundaries) and the "Other" segments. We now see from Fig. 18 that the notion of "partition" is just as important as the notion of "segment"; indeed, the two notions cannot be separated.

Referring back to Fig. 1, the MDT Data Abstraction Intelligence (DAI) subsystem 14 sits between the Client subsystem 12 and the Data and Schema Manipulation (DSM) subsystem 16. The DAI 14 (and the DSM 16) reside on the Applications Server 32 between the Desktop 30 and the Data Warehouse 24. The DAI 14 is responsible for instantiating user selected InfoFrames™ and managing several kinds of Metadata™ used in this instantiation. This Metadata™ represents (1) Dimensions and Measures that provide a customizable "dimensionalization" of the relational data in the warehouse; (2) Measure Relationships that provide explanatory text in the InfoFrame™; and (3) Metadata™ related to time. The DAI 14 also processes updates to this Metadata™ that originate in the Client layer 12 and handles several other kinds of user updates, primarily by passing them through to the DSM 16 subsystem. Finally, the DAI 14 executes Alert requests from either the Client 12 or the Scheduler subsystem 18.

The design of the overall Client/Server envisions two kinds of DAI subsystems 14. The first kind of DAI 14 is needed to handle continual, synchronous requests for information from the Client 12. For example, the Client 12 may make a number of requests for information during an active MDT session; the S-DAI (for Serial DAI) will handle these requests.

The second kind of DAI 14 will execute an Analyst, which may contain a trigger definition or an InfoFrame™ definition. This execution may take a relatively long time (e.g., hours) and will execute concurrently. This type of DAI 14 may be called the C-DAI , for Concurrent DAI. The C-DAI is spawned by an S-DAI in response to an InfoFrame™ request and given all the information it needs. When it is finished, it will write the resulting InfoFrame™ into a server disk file.

Fig. 19 shows the general, high-level data flow between the DAI 14 and the other subsystems and components of the present MDT invention.

When a user logs onto the present invention, a Serial DAI (S-DAI) 14A is created to service various kinds of requests. All user requests from the Client 12 flow through the Client/Server module (CSM) 1404, which controls communication between

Client 12 and Server 32. Requests for Metadata™, both read and update (write) requests, are handled by the S-DAI. Read requests cause the S-DAI to query the Data and Schema Manipulation (DSM) module 16 for Metadata™ contained in the MDT Metadata™ tables. The S-DAI handles update requests by using the "Classic subsystem", a system for managing heirarchical data structures available from Bell Laboratories of Lucent Technologies. The Classic subsystem may be used to check and properly position new user segments in the segment hierarchy. The Serial DAI also handles requests for previously generated InfoFrames™, fetching them from the Server disk.

When a user requests that an Analyst be generated, a Concurrent DAI 14B is created by the Dispatcher (if resources permit), and the Concurrent DAI 14B is passed all the information in the Analyst definition. The Concurrent DAI 14B then uses its built-in algorithms and Metadata™ requested from the CSM 1404 to generate the InfoFrame™ which is then stored on the Server disk.

The CSM module 1404 provides the low level services that pass messages from one process to another. Certain classes may be built on top of the CSM 1404 that make it easier to use, as described in further detail below.

The message-passing design consists of 5 classes and an enumerated type. mdt_MessageType, the enumerated type has a unique value for each type of message in MDT; mdt_TInStreamHandle and mdt_TOutStreamHandle are handles for incoming and outgoing message streams respectively; mdt_Message is an abstract base class for all MDT messages; dai_MessageHandler is an abstract base class for objects that can handle messages; finally, dai_MessageRegistry is the registry of message handlers.

mdt_MessageType is defined below.

```
enum mdt_MessageType {
   UNDEFINED,
   CS_LOGIN,
   SC_LOGIN_RESPONSE,
```

```
    CS_GENERATE_INFOFRAME,
    CS_GET_INFOFRAME_STATUS,
    SC_INFOFRAME_STATUS,
    CS_GET_INFOFRAME,
    SC_INFOFRAME,
    END_OF_ENUM
};
```

mdt_MessageType is an enum than defines all the message types understood by MDT processes. For each class derived from mdt_Message, there must be at least one value in mdt_MessageType. This declaration of mdt_MessageType is used by all components of MDT. The message types shown are only a sample. The message types currently defined below are only a sample. This definition will evolve as MDT code is written.

mdt_TInStreamHandle is defined below.

```
class mdt_TInStreamHandle : public csm_InStreamHandle {
 public:
   mdt_TInStreamHandle() : d_mtype(UNDEFINED) {}
   virtual void Connect(RWvistream *);
   mdt_MessageType GetMessageType() const;
   private:
      mdt_MessageType d_mtype;
};
```

mdt_TInStreamHandle is a typed handle for incoming (received) message streams. An mdt_TInStreamHandle is used to pick up an incoming message stream from the CSM module 1404. The mdt_TInStreamHandle automatically reads the message's type

from the incoming message stream and provides a member function to get that message type. Users of mdt_MessageRegistry need not use this class.

mdt_TOutStreamHandle is defined below.

```
class mdt_TOutStreamHandle : public csm_OutStreamHandle {
 public:
   mdt_TOutStreamHandle(mdt_MessageType t) : d_mtype(t) {}
   virtual void Connect(RWvostream *);
   mdt_MessageType GetMessageType() const;
 private:
   mdt_TOutStreamHandle();
   mdt_MessageType d_mtype;
};
```

mdt_TOutStreamHandle is a typed handle for outgoing message streams. An mdt_TOutStreamHandle is used to send a streamed message out via the CSM 1404. The mdt_TOutStreamHandle has a message type and automatically writes it's message type to the stream so that the mdt_TInStreamHandle on the receiving end can decode the message type.

mdt_Message is defined below.

```
class mdt_Message {
 public:
   mdt_Message() : d_restoreVersion(0), d_restoreFlag(false) {}
   mdt_Message(const mdt_Message &);
   virtual mdt_MessageType GetMessageType() const;
   virtual int GetLatestVersion() const = 0;
   virtual void SetRestoreVersion(int restoreVersion);
```

```
    virtual int GetRestoreVersion() const;
    virtual void saveGuts(RWvostream &) const;
    virtual void restoreGuts(RWvistream &);
    virtual bool IsRestored() const;
  protected:
    virtual void SetRestoredFlag(bool);
  private:
    bool d_restoreFlag;
    int d_restoreVersion;
};
```

Requests and Replys are communicated between Client 12, Serial DAI 14A, Concurrent DAI 14B, Scheduler 18 and Dispatch 2513 by the Client/Server Module (CSM) 1404.

CSM 1404 implements an mdt_Message class which will be used for sending or receiving Requests or Replies. A receiving mdt_Message will contain an input stream. A sending mdt_Message will contain an output stream. A stream is a well known C++ construct which allows the user to 'stream' the elements of a long message into a buffer, or to stream the elements of the message out of a buffer.

Each of the Request and Replies implemented for the present MDT invention are represented by a message derived frmo the mdt_Message base class and has appropriate functions for streaming itself into or out of a stream.

The CSM 1404 implements a csm_SimpleSocket and a csm_ServerSocket class. Each type of socket contains a TCP/IP socket. A TCP/IP socket is a well known API to a TCP/IP network. Other implementations are envisioned. Each type of socket can extract the stream buffer from a message and send it via the socket to a recieving csm_Simple or csm_ServerSocket, or can receive a stream buffer from a sending csm_SimpleSocket and install it in a message.

A csm_SimpleSocket is used to communicate messages between MDT subsystems in a committed, one to one relationship. A csm_ServerSocket is used to allow an MDT subsystem to accept messages from many other subsystems.

In Fig 25, the Client subsystem 12 will maintain a csm_SimpleSocket which it will use to request a SDAI subsystem 2512 from the Master subsystem 2511, and which it will use thereafter to exchange messages with that SDAI subsystem. The Client 12 will use this socket whenever a user input to the Client 12 requires an exchange with the Server.

In Fig 25, the Master subsystem 2511 will maintain a csm_ServerSocket to receive messages from any Client subsystem 12 which wants to request an SDAI subsystem 14A. Excepting when the Master is activly starting or otherwise tending to the SDAI's, it will be listening at the csm_ServerSocket.

The SDAI subsystem 2511 will maintain a csm_SimpleSocket to exchange messages with the Client subsystem 12. Except when it is actually implementing a Client request, the SDAI 14A will be listening at its csm_SimpleSocket.

When the user 1405 submits an Analyst for immeadiate execution, the SDAI subsystem 14A will construct a new csm_SimpleSocket to communicate the Analyst to the Dispatcher 2501. When the user sumbits a Scheduled or Triggered Analyst, the SDAI will construct a csm_SimpleSocket to communicate the Analyst to the Scheduler 18.
The Scheduler 18 will maintain a csm_ServerSocket to collect Scheduled or Triggered Analysts from any and all SDAI subsystems 2511. When the Scheduler determines the the time has come to launch a Scheduled or Triggered Analyst, it will construct a csm_SimpleSocket to communicate that Analyst to Dispatcher subsystem 2513. Except when it is testing its Scheduled and Triggered Analyst lists or dispatching them, the Scheduler will be listening at its csm_SimpleSocket.

The Dispatcher subsystem 2513 will maintain a csm_ServerSocket to collect Analysts for execution from any ready SDAI subsystem 2511 or from the Scheduler subsystem 18; or to collect Analyst requests from the CDAI subsystem 14B. When an

SDAI or the Scheduler presents an Analyst, the Dispatch will hold it until resource become available for its execution. When resources are available the SDAI will start a CDAI 14B. When the CDAI returns a request for the Analysts, the SDAI will create a csm_SimpleSocket to communicate that Analyst to the CDAI. Except when it is starting managing CDAIs, the SDAI subsytem 2513 will be listening at its csm_ServerSocket.

The CDAI subsystem 14B will construct a csm_SimpleSocket shortly after it is started to collect its Analyst from the Dispatcher subsystem 2513. After collecting this message, it will discard the csm_SimpleSocket. The CDAI 14B will exchange no other messages with the other subsystems of the present invention.

The mdt_Message abstract base class defines the object that holds the content of an MDT interprocess message. Each message has a message type, a message version, and the ability to read/write its data from/to a stream of message data. For each type of message, there is a concrete implementation of a class derived from mdt_Message. Each message class must implement GetLatestVersion to return its version and GetMessageType to return its mdt_MessageType. It must also implement the Rogue Wave saveGuts and restoreGuts methods to write its persistent member data to a stream and read the member data back from stream. Unpacking order is first in first out. There is only one derived message class per message type, but there can be several message types used by a derived message class. The same message code should be linked into both the sending and receiving processes. Version checking is used to robustly handle mismatches between the version of the code in the sending process and the receiving process. The message version is used by restoreGuts to insure that an incoming message stream can be restored and to migrate streams saved by older versions of the class to the current version. An mdt_Messages is sent/received from the CSM module 1404 by saving/restoring its guts to/from the stream pointed to by the mdt_TOutStreamHandle/mdt_TInStreamHandle class defined above.

dai_MessageHandler is defined below.

```
class dai_MessageHandler {
 public:
   virtual bool HandleMessage(const mdt_Message &) = 0;
};
```

dai_MessageHandler is an abstract base class for message handler classes. If the process uses the message registry to dispatch received messages, at least one concrete message handler class must be implemented. As many as one message handler per registered message may be implemented..

dai_MessageRegistry is defined below.

```
class dai_MessageRegistry {
  public:
    dai_MessageRegistry( csm_Connect &);
    void DispatchMessages();
    void RegisterMessage(mdt_MessageType,
                         dai_MessageCreateFunc,
                         dai_MessageHandler &);
  private:
    dai_MessageCreateFunc d_createFunc;
    dai_MessageHandler *d_handler;
    csm_Connect *d_connect;
};
```

dai_MessageRegistry is a class meant to be instantiated only once in each process that uses it. The message registry provides a method to register a handler for each message type and a method to dispatch all incoming methods. The dispatch method acts as the application's main loop. The return value of the HandleMessage method of the handler

determines whether DispatchMessages blocks or returns after a message is processed. If the return value is true, it blocks.

The following is the set of events that occur as a message is transmitted from one process to another using the MDT typed stream handles and the message registry.

1. sending process constructs an instance of a concrete message class (derived from mdt_Message), MS and loads it with the proper message data.
2. sending process creates an mdt_TOutStreamHandle object with the same message type as the message. The mdt_TOutStreamHandle object writes the message type to the stream.
3. sending process uses the message's saveGuts member function to write the message to the message stream.
4. message's saveGuts method first calls base class method which writes the message's version number to the stream. Next it saves the message's persistent member data fields to the stream.
5. sending process calls csm_Send*Process*Connect::Send() to send the message stream.
6. The CSM extracts bytes from the stream object and sends the bytes to the receiving process.
7. When the mdt_TOutStreamHandle object is destroyed, it in turn destroys the stream object it was connected to.
8. The receiving process should be waiting in a call to csm_Receive*Process*::Receive(). Internally, there is probably some queuing of messages.
9. The CSM in the receiving process gets the oldest queued message. It then converts the bytes into a stream object and connects that stream to the mdt_TInStreamHandle object that was passed to csm_Receive*Process*::Receive().

10. When the stream is connected to the handle, the handle reads the message type from the stream and remembers it.
11. Finally, control returns from csm_Receive*Process*::Receive() to the caller.
12. The receiving process gets the message type from the mdt_TInStreamHandle and constructs an empty instance of the right type of message class for that message type. If the message dispatcher is in use, this is handled by dai_MessageRegistry::DispatchMessages().
13. The receiving process calls the message's restoreGuts function. If the message dispatcher is in use, this is handled by dai_MessageRegistry::DispatchMessages().
14. The message's restoreGuts method first calls the restoreGuts method of its base class (usually mdt_Message) which reads the version number and saves it as the RestoreVersion member. Next control returns to the derived version of restoreGuts. It calls GetRestoreVersion and uses the resulting version number to determine which data members to read from the stream and what order to read them in. Next the data fields of the message class are read back from the stream.
15. The received message is now in its final form. If the dispatcher is in use, it looks up the handler for this message type in the registry and calls its HandleMessage method.
16. When the mdt_TInStreamHandle object used to receive the message is destroyed (or reconnected by another call to csm_Receive*Process*::Receive(), the stream it was pointing to is deleted.

The present MDT invention enables or other users to monitor their data warehouse 24 by defining powerful report-generation objects called Analysts. An Analyst is an encapsulation of a generic type of analysis, customized by the user 1405 by providing a set of parameters, a schedule, and a trigger condition. The Analyst periodically checks

the trigger condition if it has a trigger condition, or periodically executes on the schedule if it has a schedule. Otherwise it executes right away. InfoFrames™ contain a variety of kinds of data and hyperlinks which can be used to run other Analysts and generate related InfoFrames™. The functionality of MDT, including Analyst definition and InfoFrame™ generation, requires an "MDT view" of the data warehouse 24 that we refer to as "MDT Metadata™" 25.

The word "Metadata™", in general refers to various kinds, of "data about data" in a database or data warehouse 24 (Fig. 1). MDT Metadata™ 25 provides a customizable view of the data not restricted by the relational structure of the database. MDT Metadata™ 25 is stored in the data warehouse 24 as a set of tables and is read by MDT present invention during start-up. Populating the Metadata™ 25 in the warehouse 24 is a key part of the MDT installation process, and an MDT Administrator will generally be needed to extend and maintain the MDT Metadata™ using knowledge of the structure of the data warehouse 24.

There are 4 main kinds of MDT Metadata™: (1) Dimensions and Attributes, (2) Segments and Partitions, (3) Measures, and (4) Measure Relationships. The present specification describes each kind of Metadata™ with a series of tables. Each table shows the column names and the types of the data values in that column (in parentheses). Some columns define MDT-oriented entities or objects, and others provide mapping information between MDT Metadata™ and the relational schema of the warehouse. Primary keys for each table are italicized (if more than one column name is italicized, the combination of those column names forms the key).

MDT Metadata™ 25 includes an explicit notion that certain data attributes can have a finite set of values. These are referred to as "enumerated attributes". For example, the present invention can represent the fact that the values for the attribute "State" are limited to a finite set of values (that is, each of the 50 States of the United States, in whatever encoding the data warehouse uses). The MDT Metadata™ tables also

refer to, for convenience, sets of values that are represented in enumerated types in source code header files. The word "enum" refers to column values of this type.

The table representation of MDT Metadata™25 is first described in 4 sections, one for each main kind of Metadata™. The following are then described: representation of time in MDT, the kinds of Metadata™ that need to be stored in source code header files, and the issue of populating the Metadata™ tables during installation is discussed.

Dimensions are the starting vocabulary for the domain: they define the high-level categories of entities. For example, in a retail domain, the Dimensions might be: Product, Market, and Time (Time is a universal Dimensions applicable to most domains and is discussed in a later section). Each Dimension has a set of attributes that can be used to describe its entities; for example, the various attributes for Product, like Department, Price, Style, and Manufacturer.

All tables in this section are set up during installation and are unlikely to change often, because they are all heavily dependent on the structure of the data warehouse 24 and the industry-specific view of the data. None of the tables in this section are generally modifiable by the end user, although occasional modification may be needed by the MDT Administrator to extend MDT Metadata™ or respond to changes in the relational schema of the data warehouse.

As shown in the following table, dimensions are represented by their name and an associated Id. The Id is used to join with other tables more efficiently. The Seg-Id is the Id of the top-level segment for this dimension, and the Comment is a comment.

7256jDim -Id (int)	Name (string)	Seg-Id (int)	Comment (string)
001	Product		
002	Market		
...			...

The following table represents all of the attributes for each Dimension. Each attribute has a unique Id (Attr-Id), a name, and the Id (Dim-Id) of the Dimension they belong to. Attributes for different Dimensions can have the same name (but they will have different Ids). MDT-Type indicates the MDT type of the attribute. Each attribute is mapped to a single table and column, and we encode the data type of the field in the database that each attribute maps to (Column-Type). All the attributes for a given Dimension can be extracted from this table using the Dim-Id field. The enum values for MDT-Type are: enum, int, float, restricted-int, restricted-float, string. The enum values for Column-Type are all those types supported by the data warehouse.

Attr-Id (int)	Name (string)	Dim-Id (int)	MDT-Type (enum)	Table (string)	Column (string)	Column-Type (enum)	Comment (string)
006	Manufacturer	001	enum				
016	Size	001	int				
0057	Region	002	enum				
017	Department	001	enum				
0099	Size	002	float				
...	...	...	...				

With respect to attributes that are enumerated types, the following table represents the legitimate values for these attributes. These values will have both a user name, like "Men's Clothing" and the name of the actual data value, like "Dept-017". The information in these tables can be partially generated by a "Select ... Distinct ..." query for the attribute.

Attr-Id (int)	Display-Name (string)	*Data-Name (string)*
0017	Men's Clothing	Dept-017
0017	Housewares	
0017	Hardware	
0057	Southern	
...	...	...

For attributes that have type integer, the following table defines the appropriate ranges of values and a "typical value" if this would be useful to present to the user. Not all integer attributes have to appear in this table, if they do not have natural ranges and typical values.

Attr-Id (int)	Min (int)	Max (int)
"age"	0	120

For attributes that have type float, the following table defines the appropriate ranges of values and a "typical value" if this would be useful to present to the user. Not all float attributes have to appear in this table, if they do not have natural ranges and typical values.

Attr-Id (int)	Min (float)	Max (float)
"income"	0	1000.00

A key part of the MDT Metadata™ 25 are a set of segments and partitions for each Dimension. A segment is a set of attribute restrictions that define a class of objects of interest. For example, "Stores remodeled less than one year ago", or "non-seasonal

store-wide promotions", or "Designer X shirts of size 14 and larger". Segments are arranged in hierarchies, one for each Dimension. The hierarchies are further organized using the concept of a partition: a set of related segments differing only by the restriction on a single attribute. Segments and partitions are represented by a set of tables that capture the segment/partition hierarchy for each Dimension and define the attribute restrictions for each table.

The following table names each segment and the Dimensions it is a part of, and provides the name and the owner of the segment. The Owner string for segments that are globally owned will be defined in a header file; here and elsewhere it is shown as "ALL". There is a so-called "top-level segment" for each Dimension with the name "All X", where X

is the Dimension name. The Num-Attrs field contains the number of attributes used to restrict this segment. For the "top-level segments", Num-Attrs will be equal to 0.

Seg-Id (int)	*Dim-Id (int)*	Name (string)	Owner (string)	Num-Attrs (int)	Comment (string)
112	001	Men's Clothing	ALL		
26	001	Men's Shirts	ALL		
14	001	Designer X Shirts	pgs		
117	001	Designer X Men's Shirts	pgs		
...	...	...	...		

The following table represents all the numeric attribute restrictions for each segment represented in the following interval notation. In the first row above, if Attr-Id 017 stands for the Attribute "Size", then this row would read: "0 < = Size < = 100"; that is, "Size is greater than 0 and less than 100". Values are represented as Strings so restrictions of attributes of other type (like float or currency) can also be represented. The enum values of Operator-1 are: greater than, less than, greater than or is, less than or is, is, is not, is between.

Seg-Id (int)	*Attr-Id (int)*	Value-1 (float)	Operator-1 (enum)	Value-2 (float)
112	017	0	"between"	100
112	018			
14				
117				
...	...	...	...	...

The following table represents all the set or enumerated type attribute restrictions for each segment. The The "Data Name" column is the same as before. For example, one might represent the segment "East Coast Cities" and define it as the set {New York, Boston, Washington, ...}. To do this, several entries, one for each city,

would appear in this table. This table can also be used to represent string attribute restrictions. The enum values for Operator are: is, is not, is in list, not in list.

Seg-Id (int)	*Attr-Id (int)*	Operator (enum)	Data Name (string)
112	019	"in list"	Seattle
112			
14			
117			
...	...	...	...

The following table defines each partition and the attribute it is defined over. The default partition name is that same as that of the attribute it is defined over. In this case, the user interface will display the partition name by appending "by-" as a prefix. The Name can be updated by the user 1405.

Prtn-Id (int)	Name (string)	Owner (string)	Attr-Id (int)
59		ALL	0059
117		pgs	0017
...		...	...

The following table and the next define the segment/partition hierarchy. This table represents the child partitions of each segment. This table can also be used to find the parent segments for a given partition.

Seg-Id (int)	*Prtn-Id (int)*
112	59
13	117
...	...

The following table represents child segments for each partition. This table can also be used to find the parent partitions for each segment.

Prtn-Id (int)	*Seg-Id (int)*
19	15
221	222
...	...

Measures are values in the data warehouse 24 that can be measured over the data. For example, Sales, Price, and Market Share are all measures in the retail domain. Different Measures are "rolled up" differently; for example, Sales over several markets are added while Market Share is averaged. The present MDT invention provides a set of Basic Measures during installation and allows the user to combine them to form more complicated Composite Measures using formulas.

The following table names the Measure, defines its rollup mechanism, and maps the Measure to a table and column. The Display Units column is for InfoFrame™ generation only. Precision is the number of digits needed to the right of the decimal point. The enum values for Rollup are: add, average, count.

BM-Id (int)	Name (string)	Rollup (enum)	Table (string)	Column (string)	Display units (string)	Precision (int)	Comment (string)
055	Sales	add					
0917	Discount factor	average					
...	...	...					

Composite measures are built by combing basic measures, certain binary and unary operations, and special keywords that indicate, for example, the target segment, the

child segments of a partition, or the sibling segments of a target segment. In addition, one can encode a set of segments to restrict a measure, using the next table. The Left-Arg and Right-Arg encode the kind of argument in "Left-M" or "Right-M", as shown here:

1. Basic measure
2. Composite measure
3. "Target segment"
4. "Parent segment"
5. Segment list: 2d argument is the Slist index (see next table).
6. Sibling segments
7. Child segments

If the operator in Op is a unary operator (count, sum, average), then only the Left-M is used. If the operator in Op is a binary operator (+, -, etc.), then both the Left-M and Right-M are used.

CM-Id (int)	Owner (string)	Name (string)	Display Units (string)	Left-M (int)	Left-Arg type (enum)	Op (enum)	Right-M (int)	Right-Arg type (enum)

If the Composite Measure references a list of segments, then the elements of the list are represented in the following table.

CM-Id (int)	*Seg-Id (int)*
17	5
17	6

The following table is used to join an attribute with a basic measure to evaluate a dimensional query. The idea is that, for each attribute (that maps to a table and a column) and for each measure (that also maps to a table and a column), you need to be able to join their two tables. This table gives the column names for the two equivalent columns (keys) that can be used in the join.

Attr-Id (int)	*BM-Id (int)*	Attr-Column (string)	BM-Column (string)
5	8	"cust_age"	"mkt_share"

The following table is used to join two attributes together to evaluate a dimensional query. That is, if the previous table (above) is not sufficient to join all attributes in a dimensional query to the measure, this table can be searched to try to find a path of attributes that can be used to create multiple joins to combine all attribute tables with all measure tables.

Attr1-Id (int)	*Attr2-Id (int)*	Attr1-Column (string)	Attr2-Column (string)
17	4	"cust_age"	"age"

A Measure Relationship is a qualitative description of causality between Measures. For example, in general, if "Shelf Space" for a product goes up, then one would expect "Sales" to go up as well. In this example, "Shelf Space" is the independent Measure and "Sales" is the dependent measure. However, Measure Relationships are more

complicated than a simple statement of causality and direction. One would not expect the above example to hold over all values of "Shelf Space" but only some range of values. Similarly, one might expect the relationship to hold only if the "Shelf Space" increased by some reasonable percent. Also, one might expect a measure relationship to hold only for some segments but not for others.

The following table defines basic Measure Relationships as follows. The value for the column I-Direction is either "direct" or "inverse", an enumerated type defined in a header file. If the value is "direct", then if the Independent Measure goes up, the Dependent Measure should go up. If the value is "inverse", then if the Independent Measure goes up, the Dependent Measure should go down. The enum values for I-Direction are: direct, inverse.

MR-Id (int)	Owner (string)	Indepedent M-Id (int)	I-Direction (enum)	Dependent M-Id (int)
019	PGS	5	"direct"	21

The following table restricts the Measure Relationship to a certain range of values of the Independent Measure. The Operation can be >, <, >=, <=, =, not=, or between. For between, both Value-1 and Value-2 are used. The enum values for Operation are: is less than, is greater than, is less than or =, is greater than or =, is, is not, between, not between.

MR-Id (int)	Operation (enum)	Value-1 (float)	Value-2 (float)
019	"between"	5	100

The following table applies only to Measure Relationships with the Change Analysis Analyst definitions. It restricts the Measure Relationship to apply only when the Independent Measure changes appropriately over the time period of the Change Analysis. The enum values for Direction are: increases, decreases.

MR-Id (int)	Direction (enum)	Value (float)	Unit (string)

The following table restricts the applicability of Measure Relations by segment. Note that if a Measure Relation applies to a given segment, it will also apply to all children segments.

MR-Id (int)	Seg-Id (int)
055	19
...	...

(a)

Time is an important part of MDT Metadata™25. At one level, we would like the user to think of Time as another Dimension; however, because Time segments can be created on-the-fly (for example, the segment "Year-to-Date"), they are represented differently. We show this representation as a set of tables, although some of the information may be defined in source code header files.

The following table defines the lowest unit of granularity of time represented in the data warehouse 24. We are assuming that all representations of time in the data warehouse uses this single unit of time. The enum values for Base Unit are: day, week, month, year.

Base Unit (enum)
Day

If a database table is used by a measure, that table must have a column for time in it. The following table lists the columns for each table that represent time, for each

basic measure.

Table (string)	BM-Id (int)	Column (string)
Transaction		Time-stamp

The following table defines the different notions of year that may be important to the user 1405. The enum values for Week Start Day are: Sunday, Monday, Tuesday, ..., Saturday.

Name (string)	Start Month (string)	Start Day (int)	Week Start Day (enum)	Comment (string)
Calendar	January			
Fiscal	May			

Jan Start (int)	Feb Start (int)	Mar Start (int)	Apr Start (int)	May Start (int)	Jun Start (int)	July Start (int)	Aug Start (int)	Sept Start (int)	Oct Start (int)	Nov Start (int)	Dec Start (int)

Quarter 1 Begin Month (int)	Quarter 1 Begin Day (int)	Quarter 2 Begin Month (int)	Quarter 2 Begin Day (int)	Quarter 3 Begin Month (int)	Quarter 3 Begin Day (int)	Quarter 4 Begin Month (int)	Quarter 4 Begin Day (int)

The Serial DAI (S-DAI) 14A (Fig. 19) is responsible for providing access to MDT Metadata™ 25 to each Client 12. When a Client 12 logs into MDT, a Serial DAI 14A is created and connected to the Client session. All Client requests for Metadata™, including update requests, are handled by the Serial DAI 14A. In addition, the S-DAI 14A provides access to the MDT Administrator, allowing him or her to configure MDT using the setup screens.

The architecture of the Serial DAI is shown in Fig. 20.

The Serial DAI 14A gets requests in the form of MDT messages from the Client 12. Each MDT message has its own "reply class" in the S-DAI 14A. Messages for Metadata™ access and simple update are handled through the Metadata™ table interface of the DSM 16 (that is, without using the Classic component 14AA -- defined further below). Requests to add or delete segments use the Classic component 14AA to validate the request and update the hierarchy as needed.

Metadata™ access and simple update services are provided to the Client 12 by the Serial DAI 14A without use of the Classic component 14AA. Whenever the Client 12 needs a particular piece of Metadata™, typically to present in the user interface, the Client 12 will send a request message to the S-DAI 14A. The S-DAI 14A will access the appropriate Metadata™ tables 25 to extract the appropriate Metadata™. It will then package it up and return it to the Client 12 in one or more messages. The Metadata™ interface automatically provides access to public Metadata™ and the Metadata™ for this particular user.

The general flow of data and control is shown in Fig. 21 and described below.

- [step 2101] The Client 12 sends a Metadata™ request message to the S-DAI 14AA. This message includes the the message type and the required parameters.
- [step 2102] The Serial DAI 14A examines the message type and determines which Metadata™ table 25 is needed.

- [step 2103] Using the parameters, the S-DAI scans the Metadata™ table(s) 25 for the correct Metadata™.
- [step 2104] The Serial DAI 14A packages the Metadata™ up.
- [step 2105] The Serial DAI 14A sends a Metadata™ message back to the Client 12.

In the unlikely event of an error, the S-DAI 14A will log the error in the server error log and return an error code to the Client 12. There may be several kinds of errors of varying severity.

New Composite Measures are added from the Client 12 using the user interface. Any syntactic or semantic checking will take place there. The syntax for Composite Measures is shown above. When the Client 12 indicates that a new Composite measure has been created, the Serial DAI will add the information to the appropriate Metadata™ table, discussed previously.

An existing Composite Measure can be updated by pulling up an existing measure in the Measure Builder screen, editing it, and hitting the Save button. The edited Composite Measure will be saved to the Metadata™ tables 25 using the same Composite Measure Id. The Client 12 will display a warning message to the user 1405 to the effect that, if this Measure is being used in an Analyst, the meaning of the results presented in the subsequent InfoFrame™ will be different.

A Composite Measure can be deleted if it is owned by the user. The Composite Measure and its formula entries will be deleted from the appropriate table, previously described. A warning message will be presented to the user 1405, warning that if this Compound Measure is used by any Analyst or other Compound Measure, the subsequent InfoFrame™ will not be generated correctly and an "analyst-time" error will occur.

Adding a new measure relationship is similar to adding a new Composite measure. All checking, if any, will take place in the Client 12. When the Client indicates

that a new measure relationship has been created, the Serial DAI 14A will add the information to the appropriate Metadata™ tables.

Modifying a measure relationship is straightforward. The appropriate tables are updated and the measure relationship id is preserved. A Measure Relationship can be deleted if it is owned by the user 1405. The Measure Relationship and its table entries will be deleted from the appropriate tables. A warning message will be presented to the user, warning that if this Measure Relationship is used by an Analyst, the subsequent InfoFrame™ will not be complete and an "analyst-time" error will occur.

The Serial DAI 14A also processes requests from the MDT administrator. These requests are used to install MDT and make occasional changes to the public Metadata™. The Serial DAI 14A must handle the flow of information from both the data warehouse 24 to the Client 12 (sending both data warehouse schema information and Metadata™) and from the Client 12 to the data warehouse 24, in this case exclusively to the Metadata™. The Serial DAI 14A is not expected to have to do any processing on the data itself, i.e., to do additional integrity or exception checking.

The data warehouse schema is passed to the Client 12 in one uninterpretted bundle and the Client 12 will interpret it. There are eight steps to setting up or installing MDT. Each step involves the Client 12 getting information from the Administrator through the user interface and passing information to the S-DAI 14A. The S-DAI 14A will update the Metadata™ tables through the DSM interface. These steps are:

1. define dimensions
2. define and map attributes to database columns
3. rename coded attribute values
4. create basic segments
5. define and map basic measures to database columns
6. review the joins between database tables
7. assign database columns as time markers
8. define year types

Classic 14AA is used to represent the user's segment and partition hierarchy, as described previously. When a user adds, modifies, or deletes a segment, the Classic component 14AA determines any and all changes that need to be made to the segment/partition hierarchy, or detects one of several possible kinds of errors. If there are no errors, these changes are then used to update the Client interface and the persistent Metadata™ tables. This is shown in Fig. 22:

- [step 2201] The Client 12 sends a segment update request, either an add, delete, or modify segment message.
- [step 2202] The S-DAI receives the request.
- [step 2203] It then calls the appropriate function in the Classic module, which uses Classic to determine all of the segment/partition changes the request induces.
- [step 2004] These changes (or an appropriate error condition) are returned; if there is an error, this is passed back to the Client with no changes to the Metadata™ tables.
- [step 2205] If there is no error, all changes are passed to the Metadata™ tables.
- [step 2206] Again, assuming no errors, an acknowledge message is sent to the Client.
- [step 2207] If the knowledge base is changed, the appropriate knowledge base file is updated.

The user's segment hierarchy is kept persistent in a knowledge base files on the Server disk. This is because it would be too time-consuming to re-create it each and every time from the Metadata™ tables. There is one knowledge base file for each dimension and each user. Each file reside in some area on the Server disk and be named "dimension.user".

The user 1405 adds a new segment by selecting an existing segment (perhaps the top level segment) and then selecting a sequence of attributes and restrictions. This sequence defines a sequence of partitions and attributes. Consider the set of attributes and restrictions of: {age > 65, income > 100K}. This segment might be called "Wealthy Seniors".

This would give rise to the following sequence, assuming the segment was defined at the top level:

All Customers

[By-Age] < = partition named by its attribute

Age>65 < = automatically named intermediate egment

[By-Income]

Wealthy Seniors < = user-defined and named segment

The order of attributes is very important. That is, the final segment, "Wealthy Seniors", could also have been defined by Income and then by Age, with the same resulting final segment. However, the automatically created intermediate segment, and the two automatically created partitions, would be different. (In this case, the first partition would be "By-Income", the intermediate segment would be "Income>100", and the second partition would be "By-Age").

The following guidelines are supported for the creating of new segments:

1. The final user-named segment is created
2. "Supporting" partitions and segments are automatically created and named but only in the attribute order generated by the user 1405
3. If the final segment is a child of any existing partition it "appears there" as well.
4. If the final segment differs by a single attribute from an existing segment, the one intermediate supporting partition will be created.
5. If any segments are classified under the new segment and differ by just one attribute, the appropriate partition is created.

To illustrate guideline 4 above, let us assume the Customer hierarchy looks like this:

All Customers
[By-Income]
Moderate-Income
High-Income

The user defines "Wealthy Seniors" as above. The hierarchy should now look like this:

All Customers
[By-Age]
Age > 65
[By-Income]
Wealthy Seniors
[By-Income]
Moderate-Income
High-Income
[By-Age]
Wealthy Seniors < = segment and partition automatically added

Likewise, when the user 1405 adds a new segment, if there is an existing segment that belongs under that segment, differing by a single attribute, the existing segment will be put under the new segment and the new partition will be created.

The segment/partition hierarchy is somewhat of a strange beast. It is rooted at the "top-level segment" for that dimension, which is a segment with no attribute restrictions (this is why it represents "ALL-X", where X is the dimension, like Customer or Product). Each segment has a number of child partitions. Each partition represents a

segmentation by a single attribute and has a number of child segments. A segment can belong to several partitions. A partition has only one parent segment.

The user 1405 creates a new segment by selecting a starting or parent segment, choosing a name for the final segment, and then entering a set of attributes and restrictions in the user interface. For each attribute restriction (in the order the user chose) a segment will be created and added to the segment/partition hierarchy. When a segment is added, partitions may need to be created both above and below the new segment.

Fig. 23 shows the addition of a segment (without loss of generality, one that differs from its parent by one attribute restriction). First, the new segment must be "fit" with a partition underneath its specified parent. If the new segment fits in an existing partition or partitions, it is added to those partitions (reference numeral 2301). If not, a new partition is created, (reference numeral 2302). As with all new partitions, other segments may fit and be added. For all other parents of the new segment (Classic will tell us all parents), we first make sure the new segment and parent differ by only 1 attribute. If so, as is true of the parent at reference numeral 2303, we again add to existing partitions (reference numeral 2304) or create a new one. (The Parent at reference numeral 2305 differs by more than one attribute. This means we don't know the order of the intervening segments, so we leave it alone.) Finally, any children segments of the new segment (like reference numeral 2306) (Classic will tell us the children of a segment) are fitted with new or existing partitions, like reference numeral 2307.

The following is a very brief sketch of the algorithm for adding a new segment to a user's hierarchy. When Classic 14AA is used, this is indicated by *underlining*.

INPUT: a parent segment and a new segment (NEW). The new segment consists of a single added attribute restriction. When the user 1405 inputs more than one attribute restriction, this basic algorithm is called several times, with names generated for the intermediate segments.

greate a temporary Classic segment for NEW

```
        If NEW is identical to an existing segment return
/* for each parent:
        check that the level difference is one
        either add the segment to an existing partition or
            create a partition, add the segment, then add additional segments
*/
        get the parent segments of NEW
        for each Parent {
            if the level difference == 1 {
                fits_flag = FALSE;
                for each child partition of this segment {
                    If the segment fits in this partition {
                        fits_flag = TRUE;
                        add the segment to the partition
                    }
                }
                if (!fits_flag) {
                    create a new partition under the parent
                    add the segment to the partition
                    for all other children of the current parent {
                        if the child fits this new partition
                        add it to the partition
                    }
                }
            }
        }
/* Now that the segment has been added to all its parents:
        For all children of the new segment:
```

```
        check to see the level difference is 1
            either add it to an existing partition or
            create one
*/
        Get the children of the segment NEW
        For each child {
          if the level difference == 1 {
              for each child partition of NEW {
                  if child fits with partition {
                      add it to the partition
                      fits_flag = TRUE
                  }
                  else {
                      create a new partition under NEW
                      add the child to it
                  }
              }
          }
```

Deleting a segment can be surprisingly tricky. That is because various changes in the hierarchy can occur when a single segment is deleted. By design, we require that children of a segment to be deleted are themselves deleted if no other partitions refer to them.

With reference to Fig. 24:

FOR ALL PARENT PARTITIONS

[2401] remove the parent/segment link

[2402] if no other segment, delete the partition

FOR ALL CHILDREN OF THE PARENT SEGMENT

[2404] see if they "fit" the partition now

FOR ALL PARENT PARTITIONS

[2405] see if they are redundant and can be "collapsed"

FOR ALL CHILD PARTITIONS

[2406] remove the segment/partition link

[2407] delete the partition

FOR ALL CHILD SEGMENTS

[2408] delete the child segment link

[2409] if they have no parent partition, DELETE

[2410] DELETE THE SEGMENT

Mention was previously made of an "other" segment (referred to herein as "OS") that represents those entities not captured by the explicit segments in the final partition. For example, in a hierarchy that looks like this:

All Customers

[By-Age]

Age > 65

[By-Income]

Wealthy Seniors

the OS under the partition "By-Income" would represent people older than 65 but whose income does not make them wealthy.

The OS will not be represented explicitly in the hierarchy. This is because its definition will change depending on what segments exist. For example, if the user added a segment called "Middle class seniors", the definition of the OS would change. Rather, the OS is implicit and its attribute restrictions can be computed by taking the restrictions of its sibling segments and negating them.

The Classic knowledge base must be initialized from the persistent Metadata™ tables 25, either when a user 1405 first logs in or when a segment update

request is received. Each dimension and its respective segments and partitions can be treated as a separate knowledge base. The initialization can be performed in two ways: (1) by making direct calls to Classic 14AA, or (2) by creating an ASCII flat file that Classic 14AA can read. The former is probably more efficient, while the latter may have advantages for debugging and retraction on error.

The general steps to create a Classic knowedge base are as follows, with reference to the table previously defined:

- For each Dimension, read from the Dimension Definition table
  - ⇒ define a dimension_filler individual for that Dimension
  - ⇒ For each numeric or string attribute
    - ◊ define the attribute role
  - ⇒ For each enumerated attribute
    - ◊ define the attribute role
    - ◊ define the primitive "filler"
    - ◊ for each enumerated attribute value
      - * define the value individual
  - ⇒ For each segment definition
    - ◊ get all the attribute restrictions
    - ◊ define the segment
    - ◊ define the segment individual
  - ⇒ For all partitions
    - ◊ create a partition individual
  - ⇒ For all Segment to Child Partition mapping
    - ◊ create the mapping by adding the partition individual to the segment individual
  - ⇒ For all Partition to Child Segment mapping
    - ◊ create the mapping by adding the segment individual to the partition individual

The modification of Metadata™, including the addition, deletion, and editing (changing) of Metadata™ by both MDT Business-level users and the MDT Administrator (or Analyst level user), will now be described.

To modify Metadata™ means to change it. In one embodiment, there are three kinds of change supported by the Serial DAI 14A and the Client interface: addition, deletion, and editing. The kinds of Metadata™ involved typically include segments, composite measures, and measure relationships. There are two kinds of users 1405 who modify Metadata™: normal (or "level" users), and the MDT Administrator or Analyst Level user, both refered to in this document as the MDTA. The MDTA can usually, but not always, be thought of as another user editing public Metadata™. We are generally not concerned here with the MDTA changing other kinds of MDT Metadata™ like the mapping and join information. The table below summarizes these combinations:

Subject	Action	Object
User MDTA	Add Delete Edit	Segments Composite Measures Measure Relationships
MDTA	Add Delete	Dimensions Attributes Basic Measures

Modification of Metadata™ is done through the Client interface, including the segment builder, measure builder, measure relationship builder, and some setup screens. The general flow of control is from the Client 12 to the Serial DAI (S-DAI) 14A, to the Classic component 14AA if segments are involved, and then to the Metadata™ tables 25. Various warnings and errors may be presented to the user 1405, as appropriate.

Modification of Metadata™ gets tricky for two reasons. First, their exist dependencies between Metadata™, for example, between segments and composite measures

that refer to a segment, or dependencies between "public" Metadata™ managed by the MDTA and various user's private Metadata™ . Second, defined analysts, either scheduled or unscheduled, may refer to Metadata™ that has been modified. Several important issues must be addressed, for example:

1. What happens when an analyst refers to missing Metadata™ , i.e., Metadata™ that has been deleted?
2. What happens when Metadata™ is deleted and other Metadata™ refers to it?
3. When the MDTA changes public segments, how are the user's segment hierarchies updated?
4. What happens when the MDTA deletes a dimension, attribute, or basic measure?

One can imagine two extreme approaches along a spectrum: one, do no checking whatsoever (a "user beware" approach), two, try to capture and prevent all potential problems. In a preferred embodiment, the present invention may be designed to be closer to the "user beware" side of the spectrum, although any other suitable design may be implemented. Given this, it is desirable to give the user 1405 warnings and information when available and to not do anything that surprises the user 1405 without warning and, if possible, confirmation.

The Concurrent DAI (CDAI) 14B (Fig. 19) is the MDT subsystem that generates InfoFrames™. Its input is an InfoFrame™ Request from the Client 12 or Scheduler subsystems 18 and its output is an InfoFrame™ containing an external HTML text report contained in a file in the User Return Area. The working of the CDAI subsystem 14B, including how it is structured and the format of the resulting reports, are provided below.

The CDAI 14B may be a UNIX™ or NT process executing on the UNIX™ or NT Server platform. It is invoked by a Dispatcher component with the command line such as:

mdtqueryengine [-c <config>] [-e <errlog>]

where the config name and the errlog name are optional Configuration file and Error Log names respectively, inherited by the Dispatcher from the Master's command line when the Master invokes it.

Certain features of the CDAI's 14B function may be determined by the value of attributes defined in a Configuration file. These attribute values specify the thresholds for Interestingness Heuristics, Localization Parameters and etc.

The CDAI 14B will generate an InfoFrame™ for the User and Error Logs for the MDT Administrator in the local text. The User's local (and language) may not be the same as the Administrator's however. For this reason, the CDAI 14B may reference two Message Catalogs to collect localized text. The Native catalog will be used as a source for localized text for InfoFrames™. The Error catalog will be used as a source of localized text for the Error Log.

The text of the generated InfoFrame™ will be localizable. The localized names of Measures, Segments and Time Periods will be provided by the Client 12 or extracted from the Metadata. Other text will be kept as parameterized strings in an MDT Message Catalog. This is known as the Native catalog. The CDAI 14B will compute the name of the Native catalog from the value of the MsgCatalogPath attribute contained in the configuration file and the ISO language name provided by the client. A separate catalog must preferrably be kept to generate Error Logs in the MDT Administrators language.

The CDAI 14B will accept one mdt_InfoFrameRequest object in an RUN_ANALYST_REQUEST message. The Request may contain an InfoFrame Definition (mdt_InfoFrameDefinition), describing the InfoFrame™ to be generated. It may specify the type of the InfoFrame™, the Segment(s) to be reported over, the Measure(s) to be reported on and the Time Period(s) to report for. In one embodiment, there are five types of Reports. These are:

- Summarization - The basic analysis of a target measure over a target segment

- Change Analysis - Summarization of the difference of a target measure over two separate time periods and over a target segment
- Measure Comparison - Summarization of the difference between two measures over a target segment
- Segment Comparison - Comparison of the same measure over two separate target segments
- Trend Analysis - Report of trends over time in the target measure and related measures over the target segment and related segments

The Request may contain an InfoFrame™ Trigger (mdt_InfoFrameTrigger). The Trigger defines a test for some condition in the data warehouse 24 and an Alert flag and may contain a "nested" InfoFrame™ Request. If the Request contains a Trigger, the CDAI 14B must only implement the InfoFrame™ Definition if the condition is true. If the condition is true, the CDAI 14B must also, if the Alert flag is true, generate an Alert and, if the Trigger contains an nested Request, execute the nested Request.

In general, the CDAI's 14B output will be an InfoFrame™ object containing a localized, extended HTML Report (see Fig. 12). The InfoFrame™ will be written into a file in the User Return Area (defined below).

The User Return Area is a directory whose path is given by the UserReturnAreaPath parameter fo the configuration file. On successful completion of the InfoFrame™, the report will be named INF.<UID>.<UNQ> where UID is a User ID and UNQ is some string which guarantees that this file name will be unique from all other file names generated for this User.

The Unique identifier will be generated by the Dispatcher when it launches the CDAI 14B and will make it available to the CDAI 14B with the InfoFrame™ request. As each Report requires a unique name, a CDAI 14B instance can only generate one Report. Where an InfoFrame™ Request specifies more than one report, when a triggered request requires an Alert Report and 'other' Reports as well as 'the' Report, the CDAI 14B

accepting the request will need to dispatch new CDAI's 14B to deal with this multitide of Reports.

These extensions will be interpreted by the Client viewer. The Report will contain either an InfoFrame™ Report, an Alert Report or an Error Report. An InfoFrame™ Report will be generated when the CDAI 14B successfully completes an InfForame definition. It will be generated by the ifgn_Report class.

An Alert Report will be generated when an InfoFrame™ Request has a trigger and that trigger evaluates to true and the Alert flag is set. It will be generated by the ifgn_AlertFrame class.

An Error Frame will be generated when the CDAI 14B is unable to evaluate a trigger or an InfoFrame™ definition and lives to tell about it. It will be generated by the ifgn_ErrorFrame class.

The content of the InfoFrame™ Report (see Fig. 12) is highly variable, depending not only on the type of analysis required but on the values of the measures encountered. The Report may be organized in a variety of ways, such as a heading, four bulleted paragraphs, a table, etc. For example:

Heading - Quotes the Analyst Description provided by the User when the Analyst is defined and names the Segments, Measures and Time Periods analysed and, if the Analyst had a trigger, the trigger terms and the time at which the trigger condition was satisfied.

- Target Segment(s) Report - A bulleted paragraph which contains:
  Target Segments - Text highlighting the results for the measure(s) over the segment(s) and time period(s) directly specified in the analyst definition. Additional measures are not reported in this section.
  Parent Contribution - A bulleted paragraph highlighting any significant contribution made by the target segment(s) to its parent segment. This section is not included when the target segment(s) is a top level segment or if the target measure(s) contains a reference to a parent segment.

Sibling Comparison - A bulleted paragraph offering interesting comparisons of the target segment(s)' value to the values of its sibling segments. This section is not included when the target segment(s) is a top level segment.
Sibling Graphs - A bar or pie chart showing the values of the sibling segments or a Line graph showing trends. This graph is not produced if there are Drill Down segments.

- Drill Down - A bulleted paragraph highlighting interesting relationships between the values of child segments in the drill down partition(s) to the value for the target segment. Drill Down partitions may be specified by the user in the analyst definition. If the user does not specify any drill down partitions, MDT automatically chooses one or more interesting partitions. See the section below on choosing drill down partitions to learn how they are automatically chosen. This section is not included when no child partition exists and no unrestricted attributes exist to create or if no existing or created partition is interesting.
  Drill Down Graph - A bar or pie chart showing the values for the Drill Down Sements or a Line chart showing trends. This graph is not produced if there are no interesting Drill Down segments.
- Formula Decomposition - A bulleted paragraph highlighting interesting contributions to the composite measure of its component measures. This section is included only when the target measure is a composite measure.
- Measure Relationships - A bulleted paragraph describing possible causes for the difference between two values of the target measure (or the difference between the target and comparison measures for the Measure Comparison Analysis). The Summarization Analyis will not contain a Measure Relations paragraph.

Table - A table showing all the measure values reported during the analysis

Segments other than the Parent, Target or Comparison segment may be marked as hyperlinks (see again, e.g., Fig. 12). The underlying HREF will contain the information required to substitute this segment as the target segment for a new analyst of the same type, for the same measure, comparisons and periods.

The Alert Report is much more straight forward. It's most interesting characteristic is the hyperlink to the Analyst Name. When the InfoFrame™ Request was defined, an Analsyst may have been defined to optionally gather more information on the event. By selecting the hyperlink, the User can launch this Analyst. The text of the Alert frame may look like:

Alert: *<Analyst Name>*
<Target Segment>
<Additional Segment 1>
...
<Additional Segment n>
<Base Time Period Description>
Alert was triggered at *<time alert triggered>*
Click here to run *<Analyst Name>* now.

Trigger is: *<Measure Name>* [*<Operator>* *<Measure Name>*],
<Measure Name> [*<Operator>* *<Measure Name>*],
...
<Measure Name> [*<Operator>* *<Measure Name>*]

An Error Report may be simpler still. It is meant to communicate exactly one statement to the User 1405, a description of the error encountered. The format may be:

Error: *<Analyst Name>*

An Error Occurred at *<time error occurred>*
<Error Text>

The CDAI 14B may report errors to a server error log. The texts of the error messages are maintained in the error message catalog as parameterized, localizable strings.

4. DSM Subsystem 16 and Scheduler Subsystem 18

The data and schema manipulation subsystem 16 and scheduler subsystem 18 are described in further detail below.

The MDT Server 32 may implement two classes of Requests for the User 1405.

- Interactive Requests; Metadata™ Fetch, Metadata™ Update, InfoFrame™ Scheduling, InfoFrame™ Status and etc. The Server will implement one Interactive Request at a time for each Client. That we handle one Interactive Request at a time is almost as interesting as that they are interactive so we will also call these Serial Requests.
- Batch Requests; Trigger Requests and InfoFrame™ Generation. The Server must be able to implement multiple Batch Requests at a time. That we must handle multiple Batch Requests at a time is almost as interesting as that they are batch so we will also call these Concurrent Requests.

Each Concurrent Request will cause one or more queries against the Database. The ODBC standard supports asynchronous queries against the database but many implementations of ODBC do not. In these implementations, each Concurrent Request will require its own process. Because putting each Request in its own process allows us to work with many more ODBC implementations and passes responsibility for managing the memory and data structures for multiple Requests to the operating system, each Concurrent Request will be get its own process. This process will be the Concurrent instance.

If Serial Requests and returning results are made to be asynchronous events, a single Server instance might be able to handle all of the Server's Serial Requests. But implementing asynchronously, while not overly difficult, is rather pointless when each Client 12 can simply be handed a new Server instance. Each Client 12 is therefore assigned a Serial instance to handle its Interactive Requests.

Its important to note that Windows NT ™ implements threads, and future revisions of the UNIX ™ operating system should as well. Threads provide an opportunity to pass responsibility for handling asynchronous events to the operating system while still managing memory in a single process.

Also, for purposes of the present invention description, it will be assumed that the Serial and Concurrent instances will be implemented by unique executables, each implementing a subset of complete Server functionality appropriate to its Requests.

With reference to Fig. 25, the Server 32 may be implemented as a collection of cooperating processes. There will be five classes of processes, as described below:

- The Master process 2511, responsible for accepting Client connections with the Server and assign the Client 12 a Serial instance.
- The Serial instance 2512, responsible for executing all of this Clients Serial Requests. These will be Metadata™ Fetches and Updates, InfoFrame™ Scheduling Requests and etc. The Serial instance will queue Scheduled InfoFrame™ Requests in the Scheduler's Queue. The Serial instance will also get the Client's InfoFrame™ Generation Requests, a Concurrent Request, but it will pass this on the Dispatcher.
- The Dispatcher process 2513, responsible for assigning a Concurrent instance to each Concurrent Request passed to it by the Serial instance or the Scheduler and for reporting the state of pending and executing Concurrent Requests.
- The Concurrent instance 2514, responsible for executing a single Concurrent Request.

- The Scheduler processes 18, responsible for passing InfoFrame™ Requests to the Dispatcher at the scheduled time.

Create relationships are indicated with white pointers 2501 from parent to child. Request relationships are indicated with black arrows 2502 from sender to recipient.

A process of the Server will share several resources in common to present a single state to the client. These are the Metadata™, the Scheduler Queue and the Return Area.

When a User 1405 makes a change to the User's Metadata™, that change must be visible to all of the User's subsequent InfoFrame™ Requests. When that User 1405 is the MDTA (Administrator) and the MDTA specifies changes to global Metadata™, these changes must be visible to all subsequent Requests.

MDT's Metadata™ will be stored on the Customer's Data Warehouse 24. Accesses to this Metadata™ will be managed by the DSM 16. In order to optimize accesses to the Metadata™, the DSM 16 will implement a shared memory, write through cache of the MDT tables.

Each User 1405 will use only a subset of Metadata™, and for practical reasons that data will be re-organized from flat tables of the database into an application specific structure. Each of the User's Serial and Concurrent instances 2512, 2514 will need to keep a copy of this subset. This image will be constructed and maintained by the DAI 14.

When a User 1405 modifies a Metadata™ item, the DAI 14 will effect the change to the local image and will command the DSM 16 it update the data warehouse 24. The DSM 16 will write this change into the shared memory cache and through the cache to the warehouse 24.

The relationships and operations on Metadata™ are illustrated schematically in Fig. 26. The light arrows 2601 represent attachments to the shared memory Metadata™ Cache 2610. The bold arrows 2602 represent the path of Metadata™.

The present MDT invention will maintain a single Scheduler Queue which will list all of the InfoFrame™ Requests scheduled to execute at some future time or to execute at regular intervals.

When the User 1405 schedules an InfoFrame™ Request through the Client 12, the Schedule Request will be passed through the Serial instance 2512 to the Scheduler 18. The Scheduler 18 will accept the Request and will place the Request in the Schedule Queue. When the User 1405 deletes a Scheduled Request the Scheduler 18 will delete the Request from the Scheduler Queue. The DAI 14 will also accept a User's Schedule Status Requests. It will satisfy them by inspection of the Scheduler Queue.

At regular intervals, the Scheduler 18 will inspect the Scheduler Queue, will identify Requests that have come due and will pass them to the Dispatcher 2513. The Scheduler 18 will then remove non-recurring Requests from the Scheduler Queue.

The Dispatcher 2513 will create a Concurrent instance 2514 to execute the Requests. A Concurrent instance 2514 is a process and expect processes should be a limited resource which must be managed by the Dispatcher 2513. Thus, an InfoFrame™ Request passed to the Dispatcher 2513 may exist in one of two states: (1) pending (waiting for a process) and (2) Executing. The Dispatcher 2513 will keep a list of Pending and Executing Requests. When the User 1405 makes an InfoFrame™ Status Request, the Client 12 will pass it to the Serial instance 2512 and the DAI 14 will implement it. It will need to collect the list of the User's Request status from the Dispatcher 2513 to complete the Status Request.

The relationships and operations on the Scheduler Queue 2710 are illustrated schematically in Fig. 27. The path of the InfoFrame™ Request is represented by the black arrows 2701. Status Information regarding scheduled Requests and Pending or Executing Requests, which must be collected by the Serial instance in response to the Client's status Requests, is represented by White arrows 2702.

Completed InfoFrames™ will be parked in the Return Area between the time they are completed and the time the Client 12 calls to collect them. The Return Area is simply a directory on the file system. It will contain a sub-directory for each user 1405.

When the Concurrent instance 2514 completes an InfoFrame™ Request it will copy the InfoFrame™ into the User's sub-directory of the Return Area. Recurrent Requests may leave many InfoFrames™ in the Return Area. The Concurrent instance 2514 must generate unique names for each Report.

The Client 12 will occasionally pass InfoFrame™ Status Requests to the Serial instance 2512. The DAI 14 will implement the Request. The DAI 14 will need to inspect the User's sub-directory of the Return Area to identify the InfoFrames™ that have been completed. The DAI 14 might 'decode' the file names to produce Analyst names and execution dates to report to the Client 12.

The Client 12 will also occasionally pass InfoFrame™ Upload Requests to the Serial instance 2512. The DAI 14 will implement the Request. The DAI 14 will collect the InfoFrame™ from the User's sub-directory of the Return Area. It will pass the InfoFrame™ to the Client 12 and will remove the image from the Return Area when the Client 12 acknowledges receipt.

The relationships and operations on the Return Area 2810 are illustrated schematically in Fig. 28. The flow of the completed InfoFrame™ is represented by the black arrow 2801. The movement of status required by the Serial instance 2512 to satisfy a Client Status Request is represented by the white arrow 2802.

1. DSM Subsystem 16

The DSM Subsystem 16 is described in further detail below.

The SQL Generator receives a Dimensional Query from the InfoFrame™ Generator, generates the necessary database queries, and returns the results to the InfoFrame™ Generator. The interface to the database is through ODBC, which takes queries in the form of SQL strings.

The SQL Generator must query the database 24 to evaluate each Measure/Segment pair. The Measures may be from the dai_MeasureList. In the first form of QueryDatabase(), the segments are the child segments of the targetPartition; in the second form, each Measure is evaluated only for the implied target segment. Each of the measures may be a Composite Measure, which may require multiple queries to evaluate the Measures that make up the Composite Measure.

Standard SQL programming techniques may be used to implement the DSM Subsystem 16.

2. Scheduler Subsystem 18

The scheduler subsystem 18 is described in further detail below.

The scheduler subsystem 18 is responsible for submitting Analysts with schedules and/or triggers to be run. It is also responsible for maintaining lists of scheduled and/or triggered Analysts; deleting, modifying and adding to those lists.

In this section, an Analyst which has a schedule, a trigger, or a schedule and a trigger will be referred to as a 'scheduled' Analyst unless there is a difference in the way the three types of Analysts behave.

When an InfoFrame™ request is received by the S-DAI subsystem 14A, it will be passed to the Scheduler subsystem 18 if it is associated with a scheduled Analyst. The Scheduler 18 will determine the proper time period in which to dispatch the Analyst. When it is scheduled to run, a request will be passed to the dispatcher 2513 as in the same manner that the S-DAI subsystem 14A passes non-scheduled requests.

Fig. 29 illustrates this process in further detail.

From the S-DAI 14A, the Scheduler 18 will receive scheduled InfoFrame™ requests, delete and disable user requests, delete scheduled Analyst requests. To uniquely identify requests, the S-DAI 14A must provide an Analyst id (contained in the InfoFrame™ request object) and a user id.

When a scheduled InfoFrame™ request is received, the Analyst is placed on the Trigger List 2901 if the Analyst has a trigger, or the Schedule List 2902 if it has either a schedule or a schedule and a trigger. The difference between a triggered Analyst and a scheduled, triggered Analyst is that the former is run at each trigger period while the latter is run only when scheduled.

The Scheduler 18 has two execution time periods, one for triggered requests and one for scheduled requests. The two time periods are configurable on each MDT server 32 and may be changed by the MDT Administrator.

When the trigger time period occurs, the Scheduler 18 traverses its list 2901 of triggered events. For those scheduled to run during that time slice and whose user account is enabled, a copy of the InfoFrame™ request without the trigger is passed to the Dispatcher 2513. An analogous process is followed for the schedule time period and schedule list 2902.

If a user 1405 is deleted, the Scheduler 18 will remove any Analysts from the lists which are owned by the deleted user. If a user 1405 is disabled, any Analysts on the lists will not run until the user 1405 is once again enabled. If an Analyst is modified, the user 1405 must explicitly remove any associated scheduled requests or they will continue to run with the old Analyst definition.

4. Brief Description of Drawings

Fig. 1 is a block diagram of the system of the present invention;

Fig. 2 is a block diagram of a client subsystem within the system of Fig. 1;

Fig. 3 is a block diagram of a data abstraction intelligence subsystem within the system of Fig. 1;

Fig. 4 is a block diagram of a data and schema manipulation subsystem within the system of Fig. 1;

Fig. 5 is a block diagram of a scheduler subsystem within the system of Fig. 1;

Figs. 6-12 are views of a tool for creating reports which employs a graphic user interface;

Fig. 13 is a flow diagram illustrating how Metadata™ is created.

Fig. 14 is another block diagram of a client subsystem within the system of Fig. 1.

Fig. 15 is yet another block diagram of a client subsystem within the system of Fig. 1.

Fig. 16 is a database heirarchy that may be created according to the teachings of the present invention.

Fig. 17 is another heirarchy that may be created according to the teachings of the present invention.

Fig. 18 is yet another heirarchy that may be created according to the teachings of the present invention.

Fig. 19 is a general, high-level data flow between the DAI and the other subsystems and components of the system of the present invention.

Fig. 20 is an architecture of a Serial DAI subsystem of Fig. 1.

Fig. 21 is a flow diagram for the Serial DAI system of Fig. 1.

Fig. 22 is another flow diagram for the Serial DAI system of Fig. 1.

Fig. 23 is a depiction of the addition of a database segment in the system of the present invention.

Fig. 24 is a depiction of the deletion of a database segment in the system of the present invention.

Fig. 25 is a description of the processes performed by the server of Fig. 1.

Fig. 26 is a diagram showing the relationships and operations with respect to Metadata™.

Fig. 27 is a diagram showing the relationships and operations with respect to the scheduler queue.

Fig. 28 is a diagram showing the relationships and operations with respect to the return area.

Fig. 29 is a diagram depicting the requests performed by an analyst.

FIG. 1

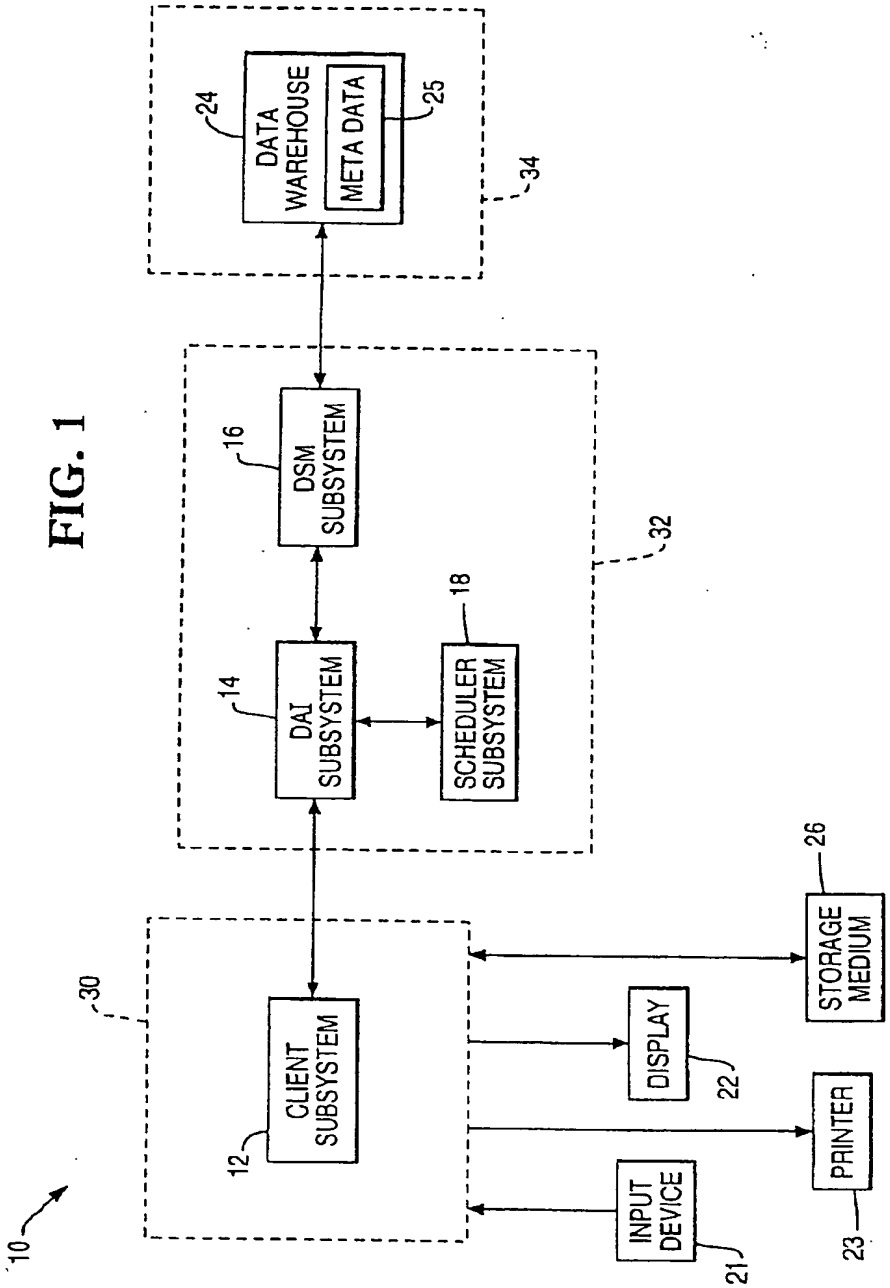

10
30
CLIENT SUBSYSTEM
12
DAI SUBSYSTEM
14
DSM SUBSYSTEM
16
SCHEDULER SUBSYSTEM
18
32
DATA WAREHOUSE
24
META DATA
25
34
INPUT DEVICE
21
DISPLAY
22
PRINTER
23
STORAGE MEDIUM
26

FIG. 2

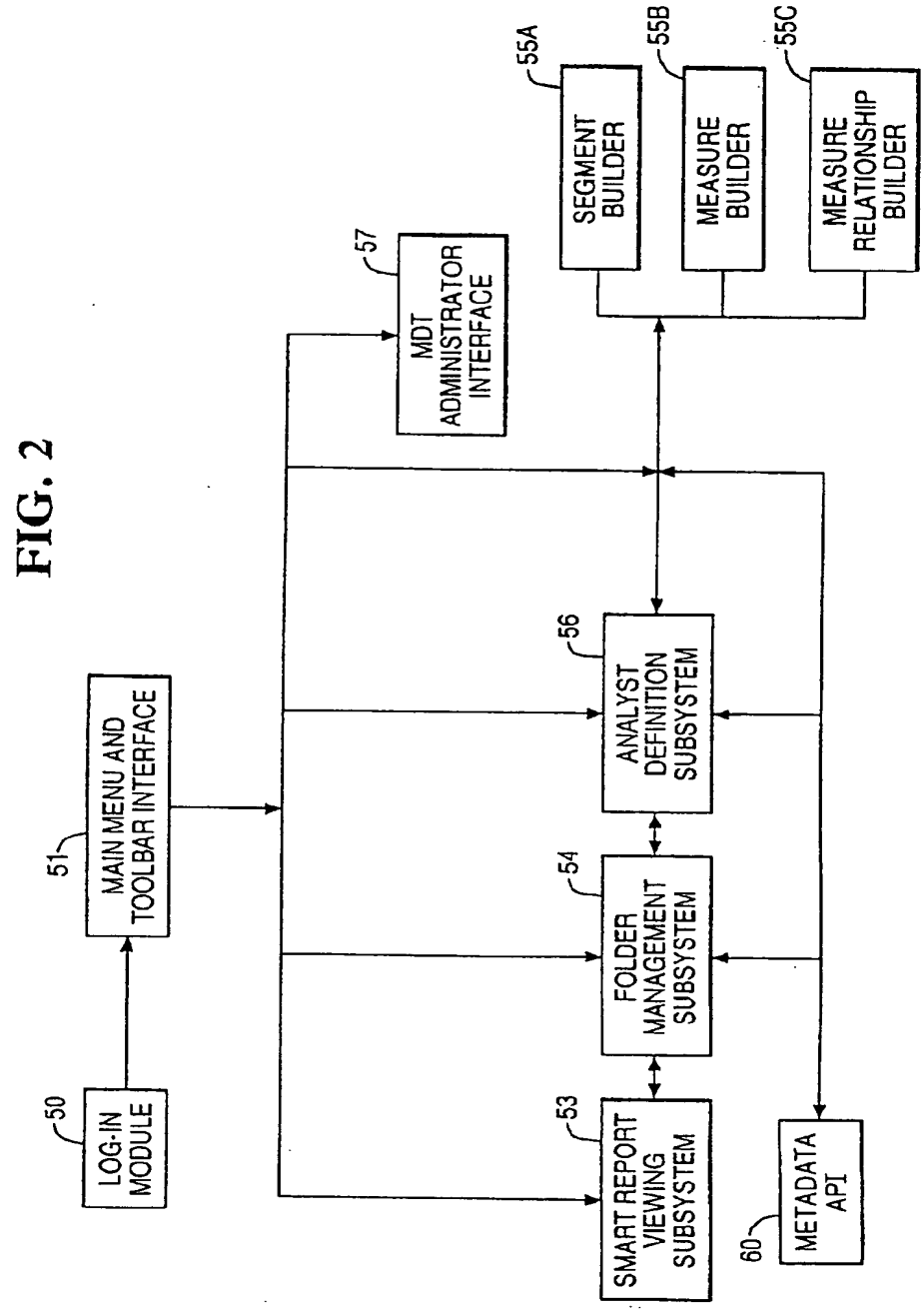

LOG-IN MODULE
50
MAIN MENU AND TOOLBAR INTERFACE
51
SMART REPORT VIEWING SUBSYSTEM
53
FOLDER MANAGEMENT SUBSYSTEM
54
ANALYST DEFINITION SUBSYSTEM
56
MDT ADMINISTRATOR INTERFACE
57
SEGMENT BUILDER
55A
MEASURE BUILDER
55B
MEASURE RELATIONSHIP BUILDER
55C
METADATA API
60

FIG. 3

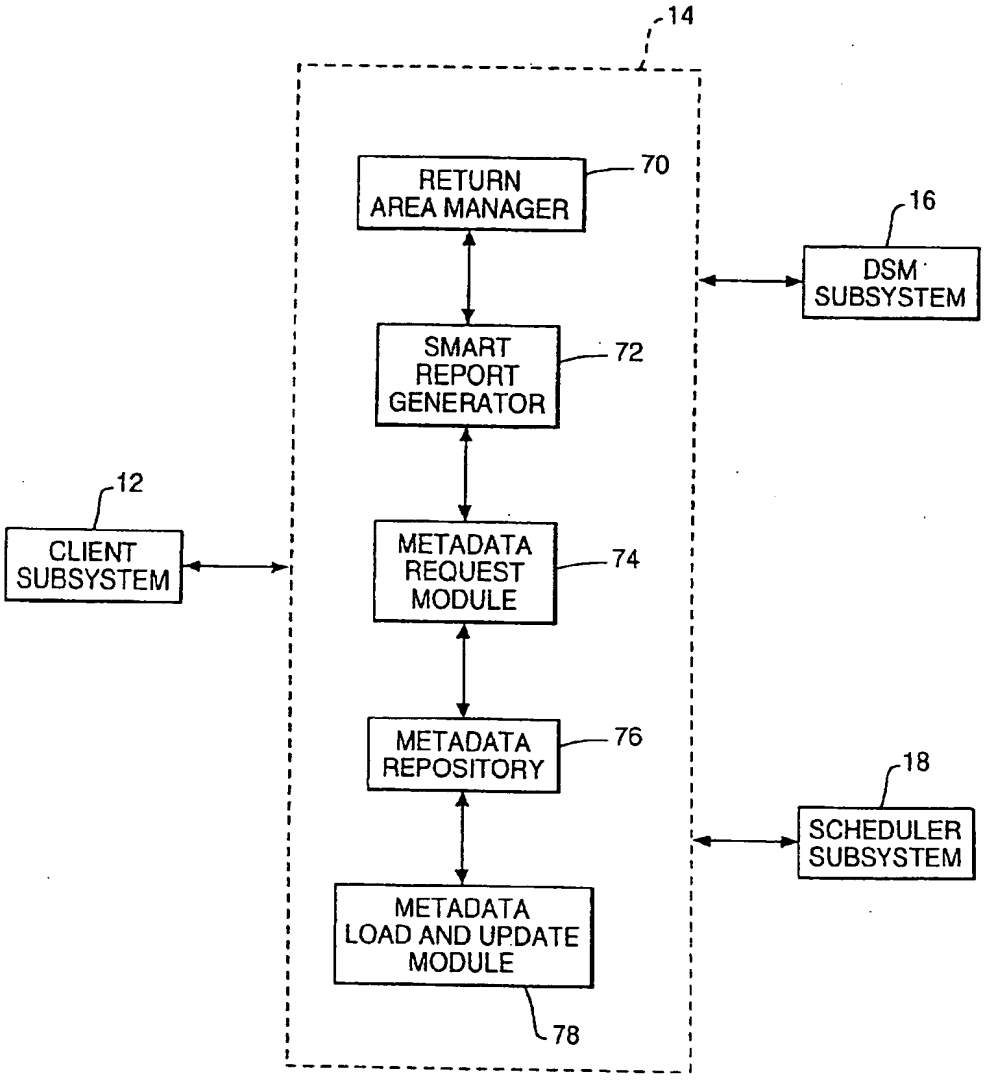

14
RETURN AREA MANAGER
70
16
DSM SUBSYSTEM
SMART REPORT GENERATOR
72
12
CLIENT SUBSYSTEM
METADATA REQUEST MODULE
74
METADATA REPOSITORY
76
18
SCHEDULER SUBSYSTEM
METADATA LOAD AND UPDATE MODULE
78

FIG. 4

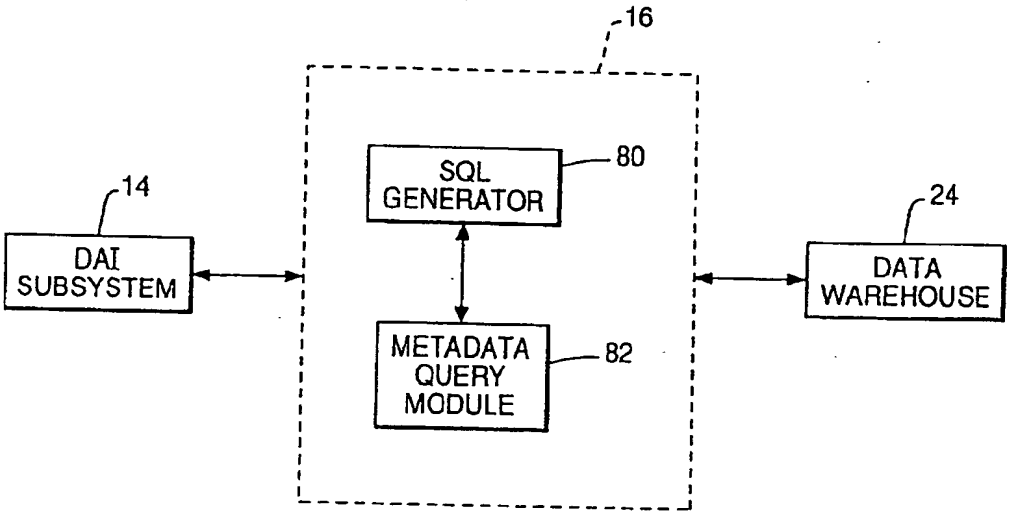

16
SQL GENERATOR
80
14
DAI SUBSYSTEM
24
DATA WAREHOUSE
METADATA QUERY MODULE
82

FIG. 5

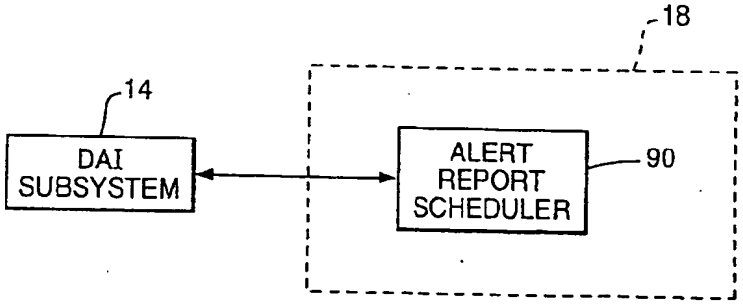

18
14
DAI SUBSYSTEM
ALERT REPORT SCHEDULER
90

FIG. 6

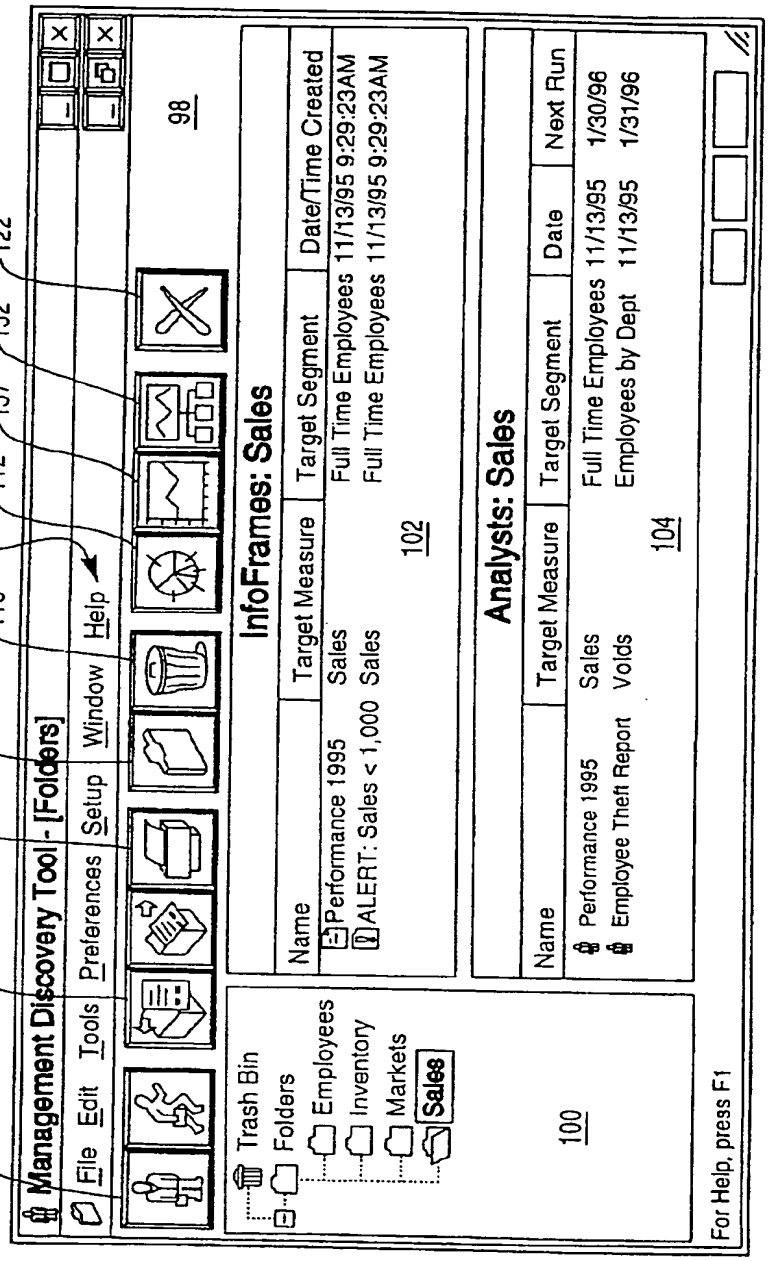

Management Discovery Tool - [Folders]
File Edit Tools Preferences Setup Window Help
Trash Bin
Folders
Employees
Inventory
Markets
Sales
100
InfoFrames: Sales
Name
Target Measure
Target Segment
Date/Time Created
Performance 1995
Sales
Full Time Employees
11/13/95 9:29:23AM
ALERT: Sales < 1,000
Sales
Full Time Employees
11/13/95 9:29:23AM
102
Analysts: Sales
Name
Target Measure
Target Segment
Date
Next Run
Performance 1995
Sales
Full Time Employees
11/13/95
1/30/96
Employee Theft Report
Voids
Employees by Dept
11/13/95
1/31/96
104
98
For Help, press F1
110
120
150
118
116
106
112
157
152
122

FIG. 7A

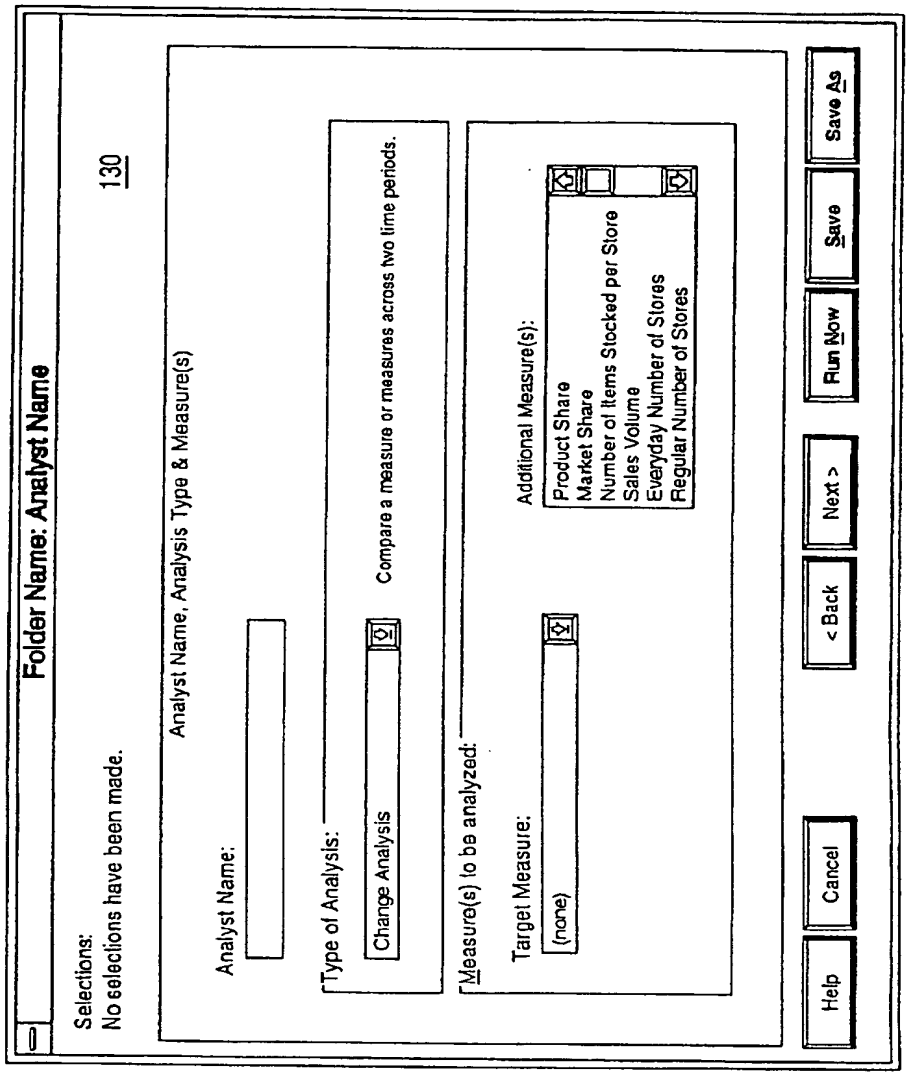

Folder Name: Analyst Name
Selections:
No selections have been made.
130
Analyst Name, Analysis Type & Measure(s)
Analyst Name:
Type of Analysis:
Change Analysis
Compare a measure or measures across two time periods.
Measure(s) to be analyzed:
Target Measure:
(none)
Additional Measure(s):
Product Share
Market Share
Number of Items Stocked per Store
Sales Volume
Everyday Number of Stores
Regular Number of Stores
Help
Cancel
< Back
Next >
Run Now
Save
Save As

FIG. 7B

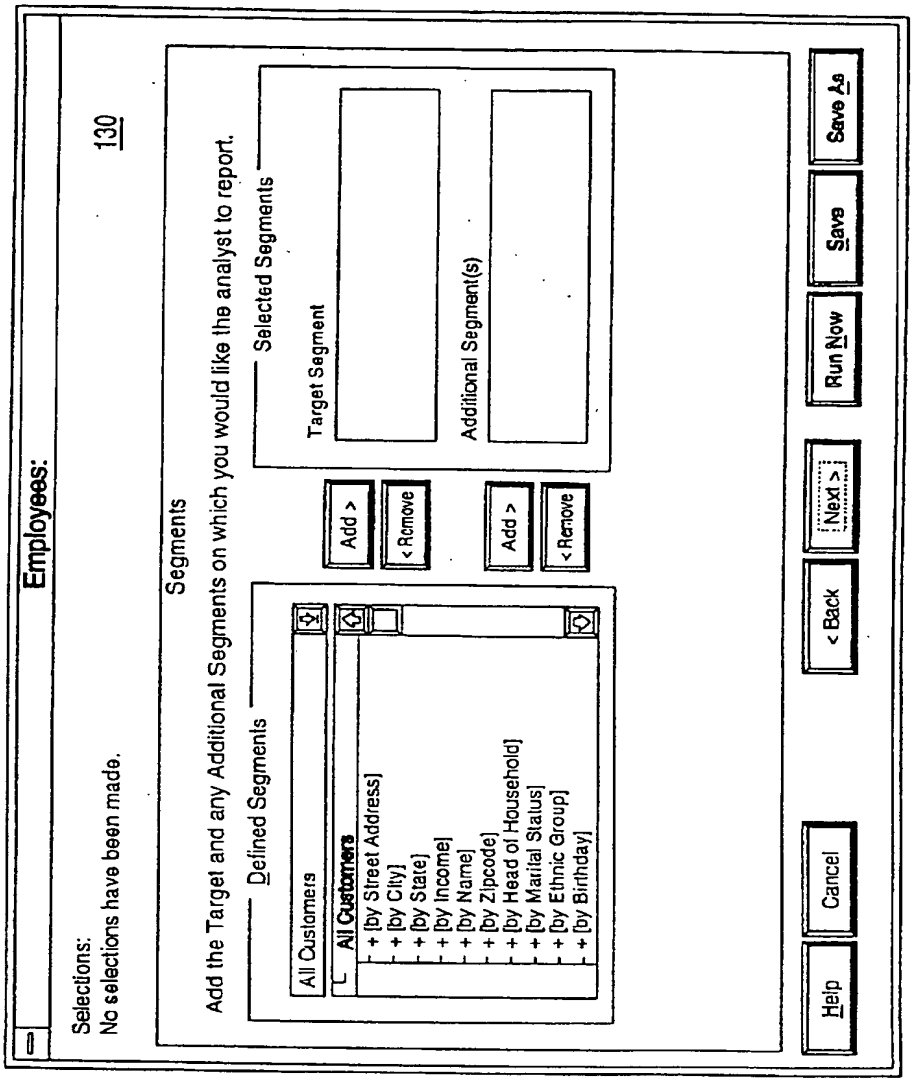

Employees:
Selections:
No selections have been made.
130
Segments
Add the Target and any Additional Segments on which you would like the analyst to report.
Defined Segments
All Customers
All Customers
+ [by Street Address]
+ [by City]
+ [by State]
+ [by Income]
+ [by Name]
+ [by Zipcode]
+ [by Head of Household]
+ [by Marital Status]
+ [by Ethnic Group]
+ [by Birthday]
Add >
< Remove
Add >
< Remove
Selected Segments
Target Segment
Additional Segment(s)
Help
Cancel
< Back
Next >
Run Now
Save
Save As

FIG. 7C

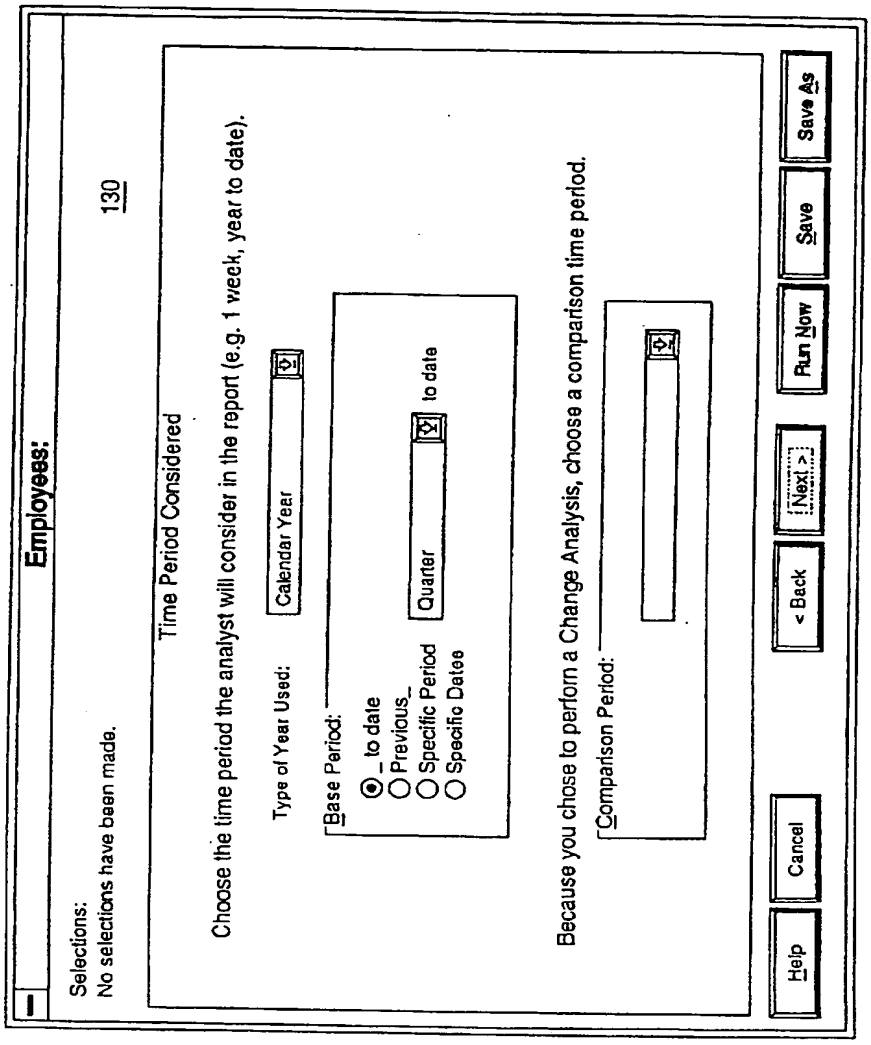

Employees:
Selections:
No selections have been made.
130
Time Period Considered
Choose the time period the analyst will consider in the report (e.g. 1 week, year to date).
Type of Year Used:
Calendar Year
Base Period:
to date
Previous_
Specific Period
Specific Dates
Quarter
to date
Because you chose to perform a Change Analysis, choose a comparison time period.
Comparison Period:
Help
Cancel
< Back
Next >
Run Now
Save
Save As

FIG. 7D

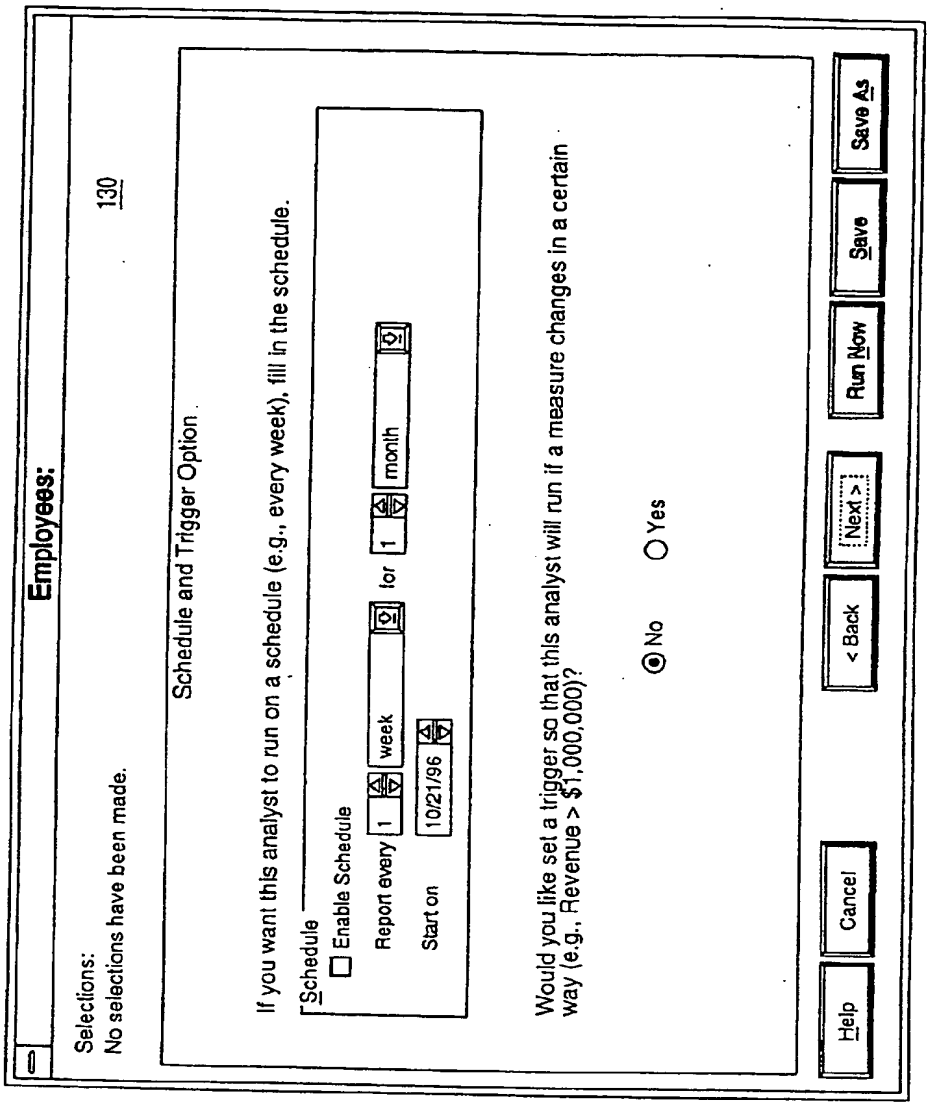

Employees:
Selections:
No selections have been made.
130
Schedule and Trigger Option
If you want this analyst to run on a schedule (e.g., every week), fill in the schedule.
Schedule
Enable Schedule
Report every 1 week for 1 month
Start on 10/21/96
Would you like set a trigger so that this analyst will run if a measure changes in a certain way (e.g., Revenue > $1,000,000)?
No
Yes
Help
Cancel
< Back
Next >
Run Now
Save
Save As

FIG. 7E

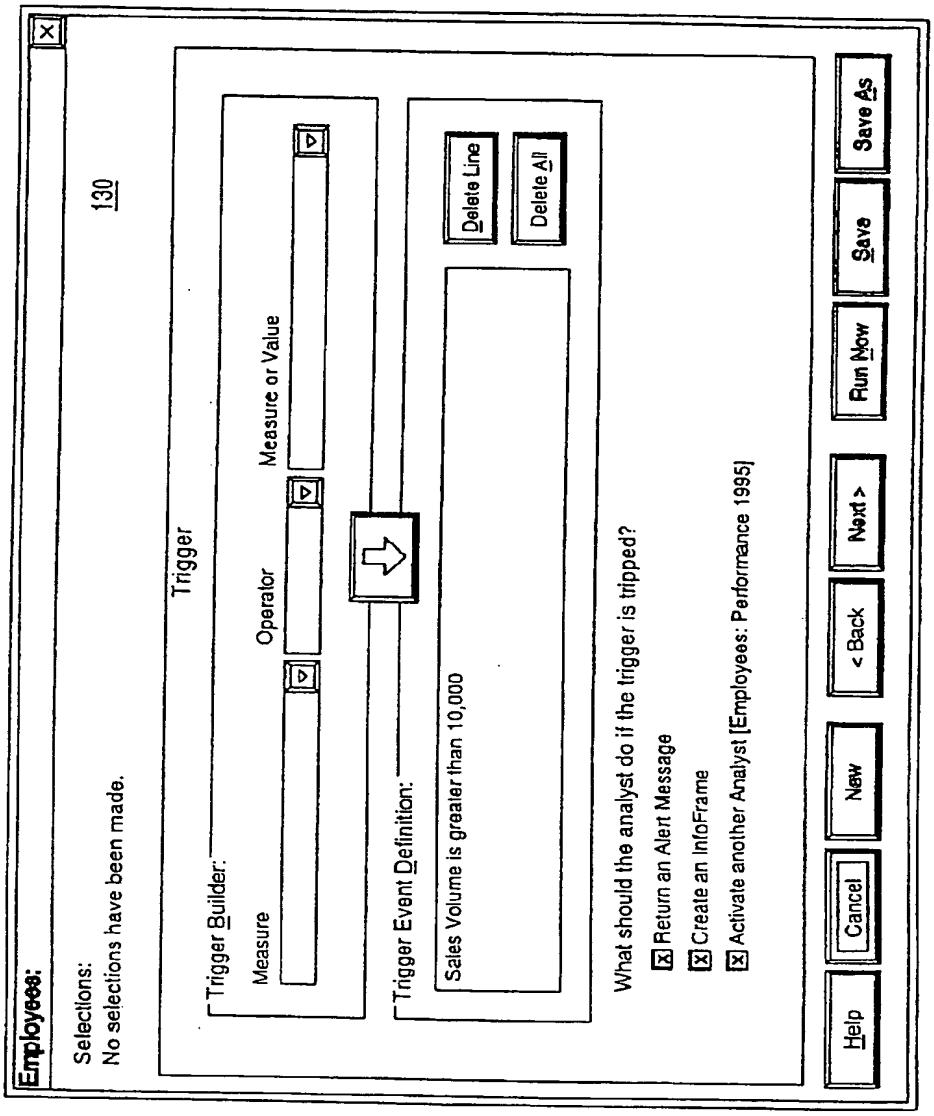

Employees:
Selections:
No selections have been made.
130
Trigger
Trigger Builder:
Measure
Operator
Measure or Value
Trigger Event Definition:
Sales Volume is greater than 10,000
Delete Line
Delete All
What should the analyst do if the trigger is tripped?
Return an Alert Message
Create an InfoFrame
Activate another Analyst [Employees: Performance 1995]
Help
Cancel
New
< Back
Next >
Run Now
Save
Save As

FIG. 8A

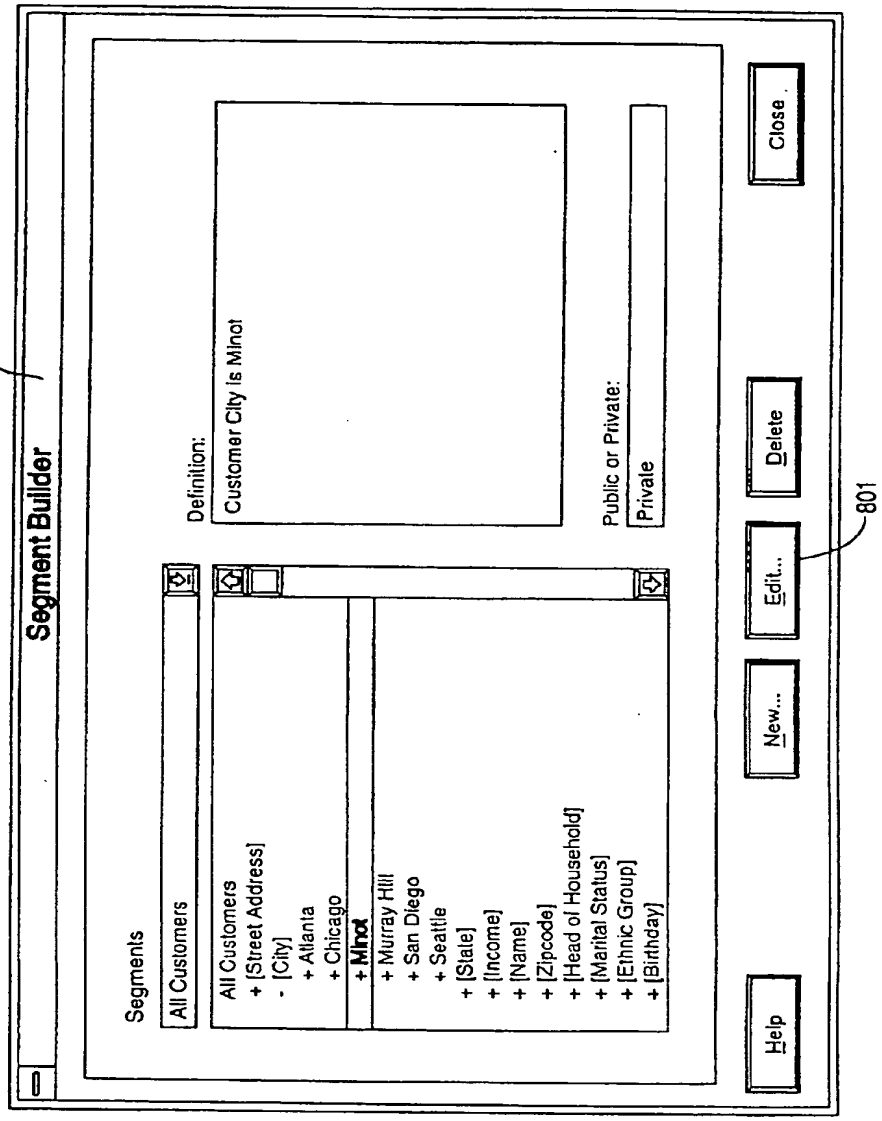

132
Segment Builder
Segments
All Customers
All Customers
+ [Street Address]
- [City]
+ Atlanta
+ Chicago
+ Minot
+ Murray Hill
+ San Diego
+ Seattle
+ [State]
+ [Income]
+ [Name]
+ [Zipcode]
+ [Head of Household]
+ [Marital Status]
+ [Ethnic Group]
+ [Birthday]
Definition:
Customer City Is Minot
Public or Private:
Private
Help
New...
Edit...
Delete
Close
801

FIG. 8B

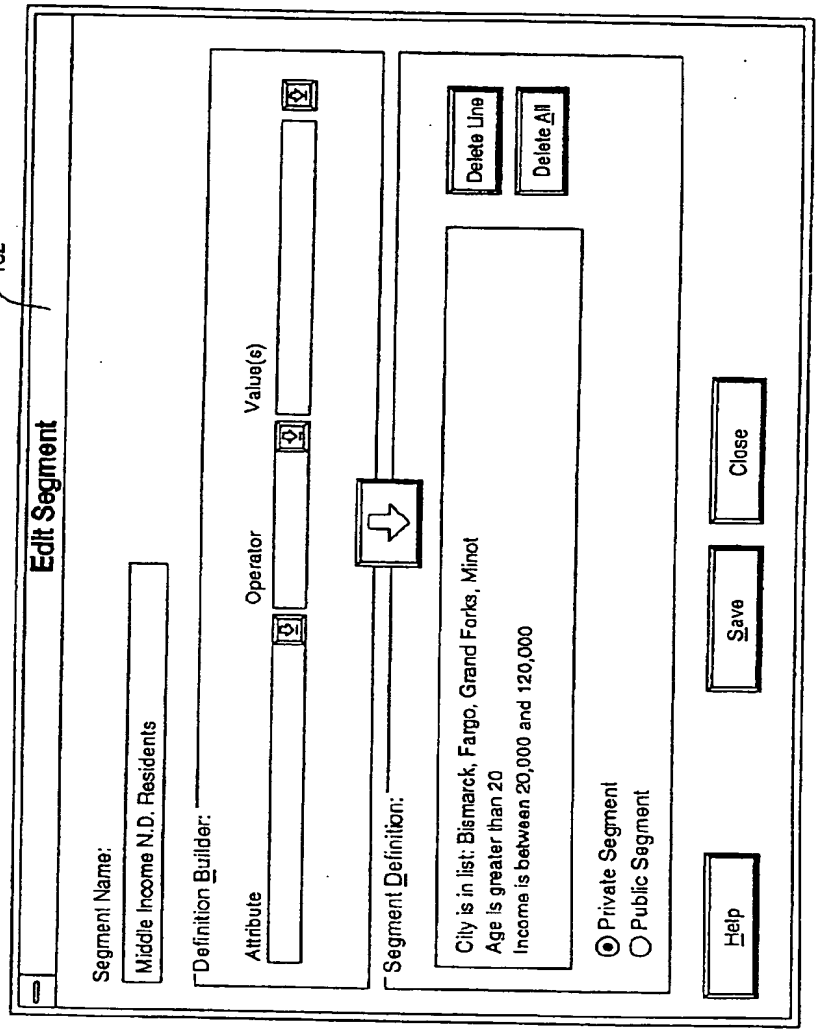

132
Edit Segment
Segment Name:
Middle Income N.D. Residents
Definition Builder:
Attribute
Operator
Value(s)
Segment Definition:
City is in list: Bismarck, Fargo, Grand Forks, Minot
Age is greater than 20
Income is between 20,000 and 120,000
Delete Line
Delete All
Private Segment
Public Segment
Help
Save
Close

FIG. 9A

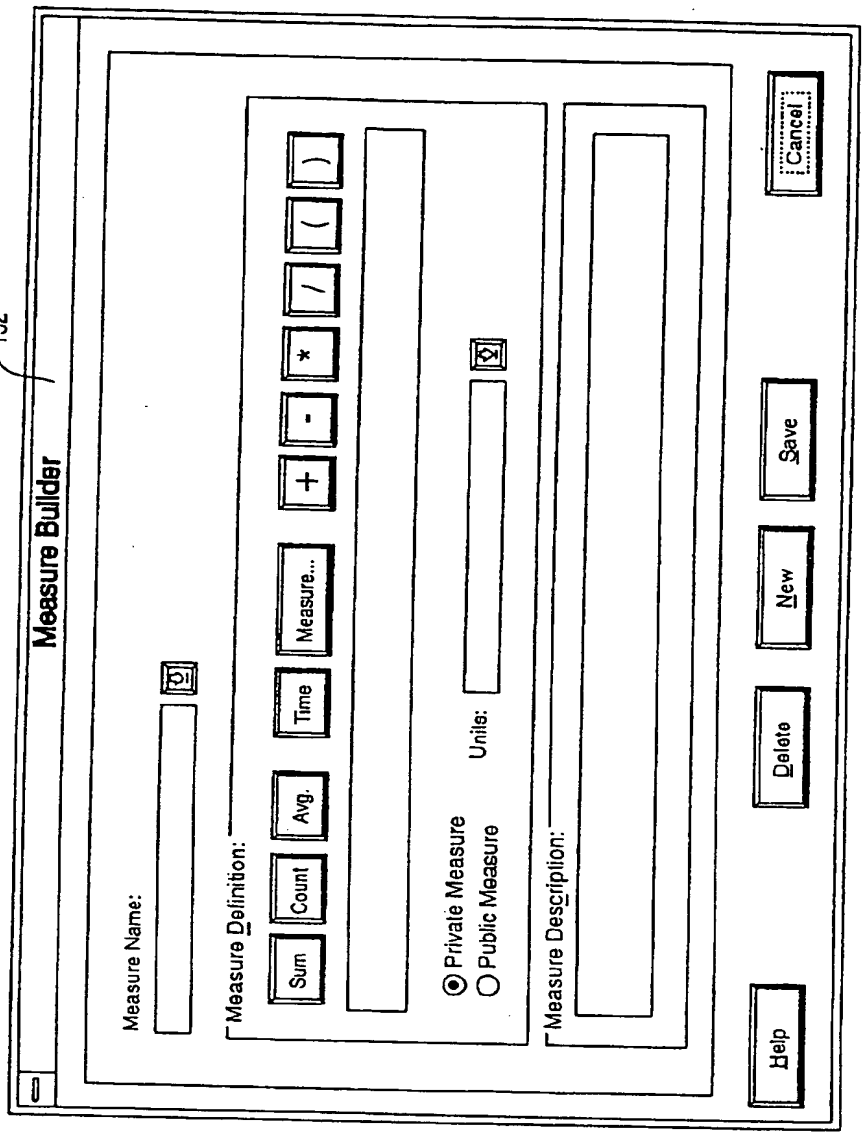

132
Measure Builder
Measure Name:
Measure Definition:
Sum
Count
Avg.
Time
Measure...
+
-
*
/
(
)
Private Measure
Public Measure
Units:
Measure Description:
Help
Delete
New
Save
Cancel

FIG. 9B

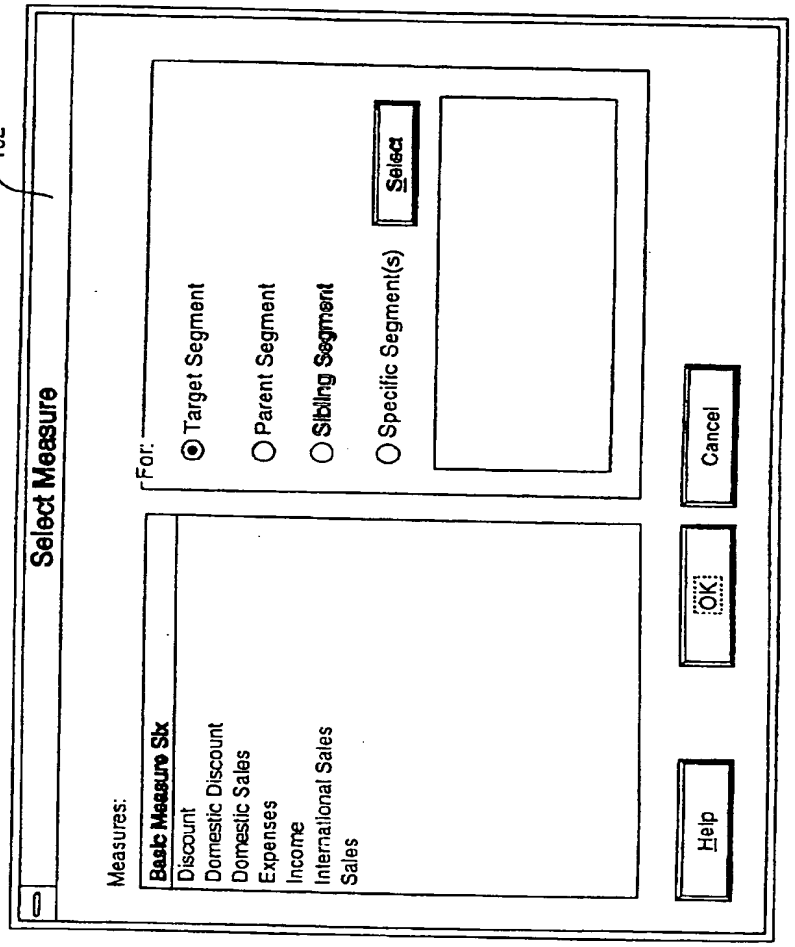

132
Select Measure
Measures:
Basic Measure Six
Discount
Domestic Discount
Domestic Sales
Expenses
Income
International Sales
Sales
For:
Target Segment
Parent Segment
Sibling Segment
Specific Segment(s)
Select
Help
OK
Cancel

FIG. 9C

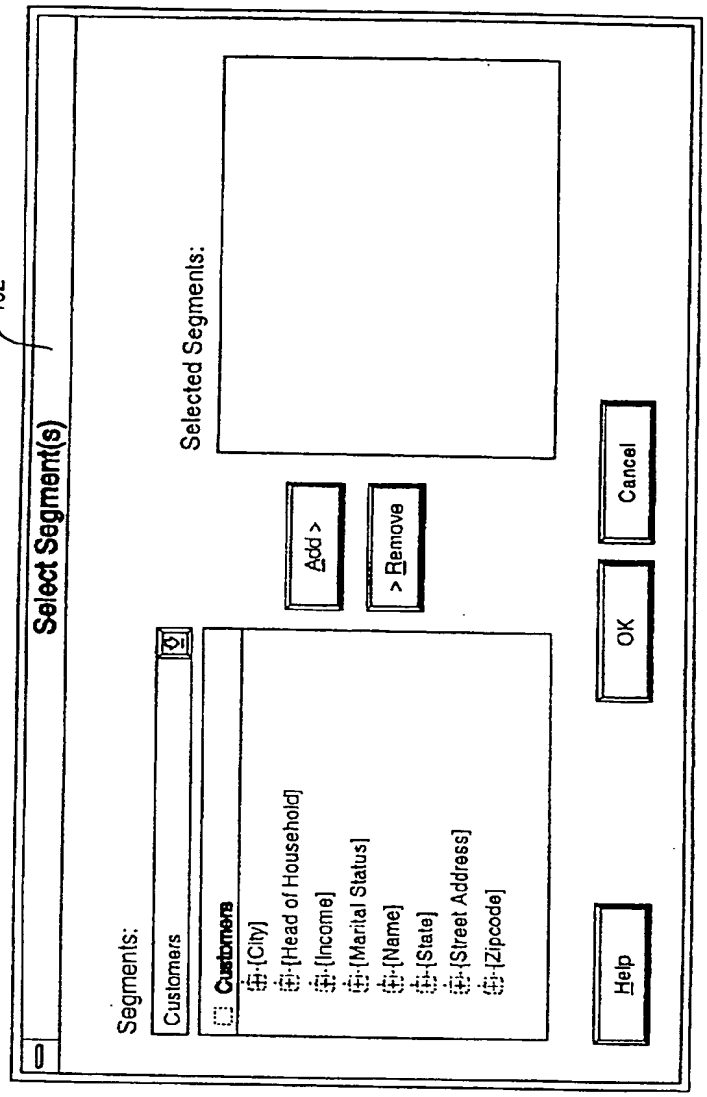

132
Select Segment(s)
Segments:
Customers
Customers
[City]
[Head of Household]
[Income]
[Marital Status]
[Name]
[State]
[Street Address]
[Zipcode]
Add >
> Remove
Selected Segments:
Help
OK
Cancel

FIG. 10

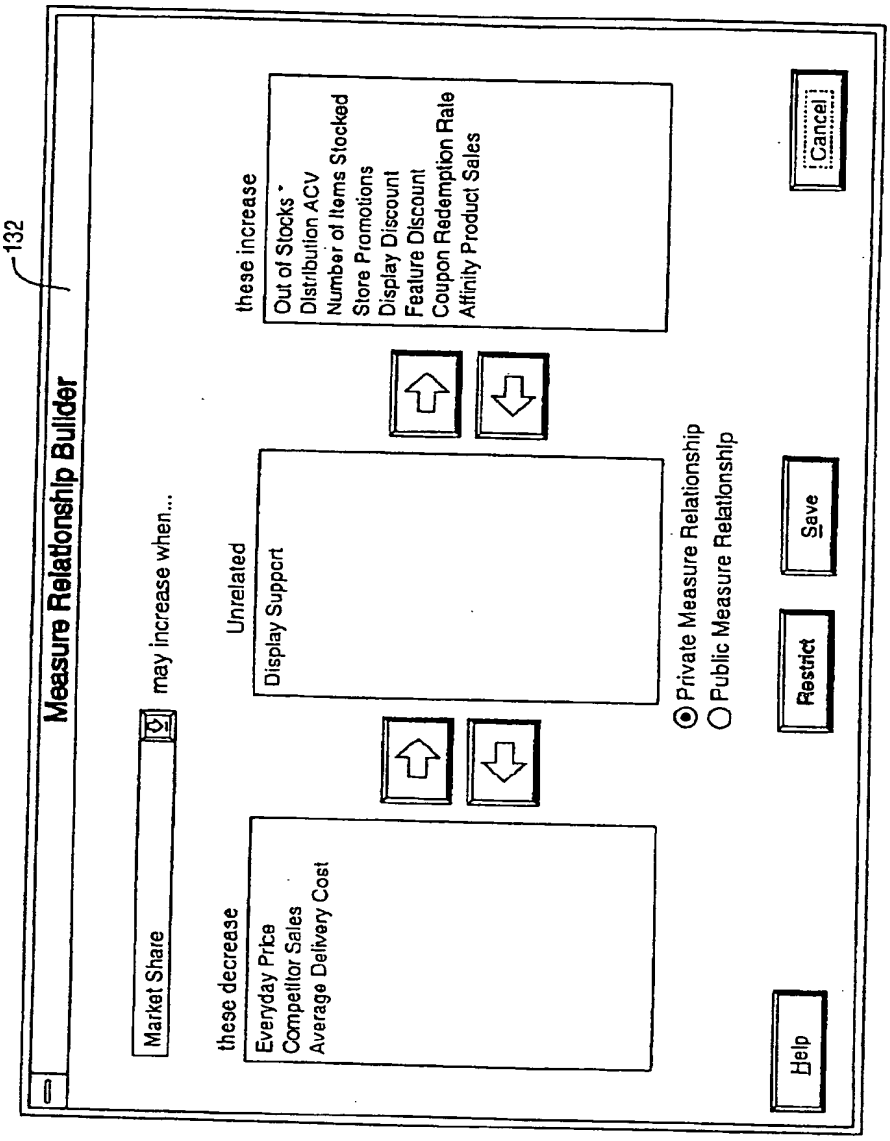

132
Measure Relationship Builder
Market Share
may increase when...
these decrease
Everyday Price
Competitor Sales
Average Delivery Cost
Unrelated
Display Support
these increase
Out of Stocks *
Distribution ACV
Number of Items Stocked
Store Promotions
Display Discount
Feature Discount
Coupon Redemption Rate
Affinity Product Sales
Private Measure Relationship
Public Measure Relationship
Help
Restrict
Save
Cancel

FIG. 11

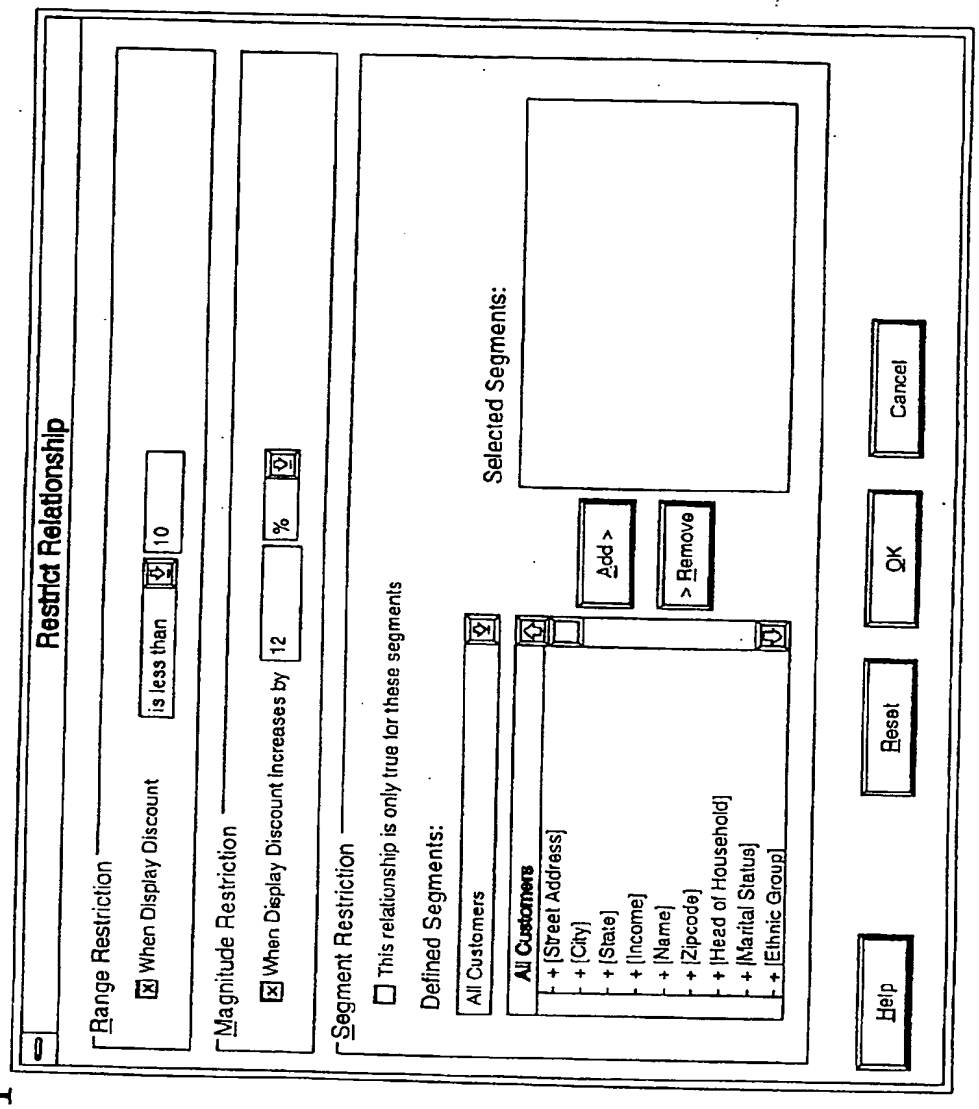

Restrict Relationship
Range Restriction
When Display Discount
is less than
10
Magnitude Restriction
When Display Discount increases by
12
%
Segment Restriction
This relationship is only true for these segments
Defined Segments:
All Customers
Selected Segments:
All Customers
+ [Street Address]
+ [City]
+ [State]
+ [Income]
+ [Name]
+ [Zipcode]
+ [Head of Household]
+ [Marital Status]
+ [Ethnic Group]
Add >
> Remove
Help
Reset
OK
Cancel

FIG. 12

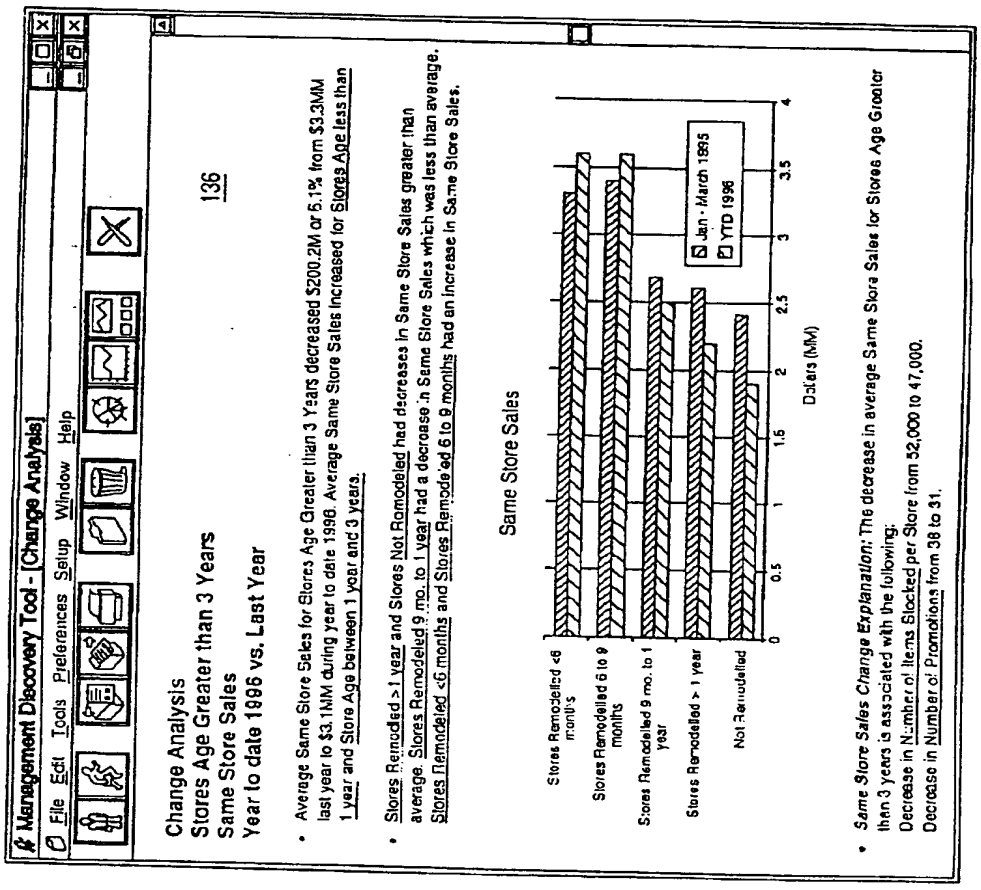

Management Discovery Tool - [Change Analysis]
File Edit Tools Preferences Setup Window Help
Change Analysis
Stores Age Greater than 3 Years
Same Store Sales
Year to date 1996 vs. Last Year
136
• Average Same Store Sales for Stores Age Greater than 3 Years decreased $200.2M or 6.1% from $3.3MM last year to $3.1MM during year to date 1996. Average Same Store Sales increased for Stores Age less than 1 year and Store Age between 1 year and 3 years.
• Stores Remodled > 1 year and Stores Not Remodeled had decreases in Same Store Sales greater than average. Stores Remodeled 9 mo. to 1 year had a decrease in Same Store Sales which was less than average. Stores Remodeled <6 months and Stores Remodeled 6 to 9 months had an increase in Same Store Sales.
Same Store Sales
Stores Remodelled <6 months
Stores Remodelled 6 to 9 months
Stores Remodelled 9 mo. to 1 year
Stores Remodelled > 1 year
Not Remodelled
Jan - March 1995
YTD 1996
0
0.5
1
1.5
2
2.5
3
3.5
4
Dollars (MM)
• Same Store Sales Change Explanation: The decrease in average Same Store Sales for Stores Age Greater than 3 years is associated with the following:
Decrease in Number of Items Stocked per Store from 52,000 to 47,000.
Decrease in Number of Promotions from 38 to 31.

FIG. 13

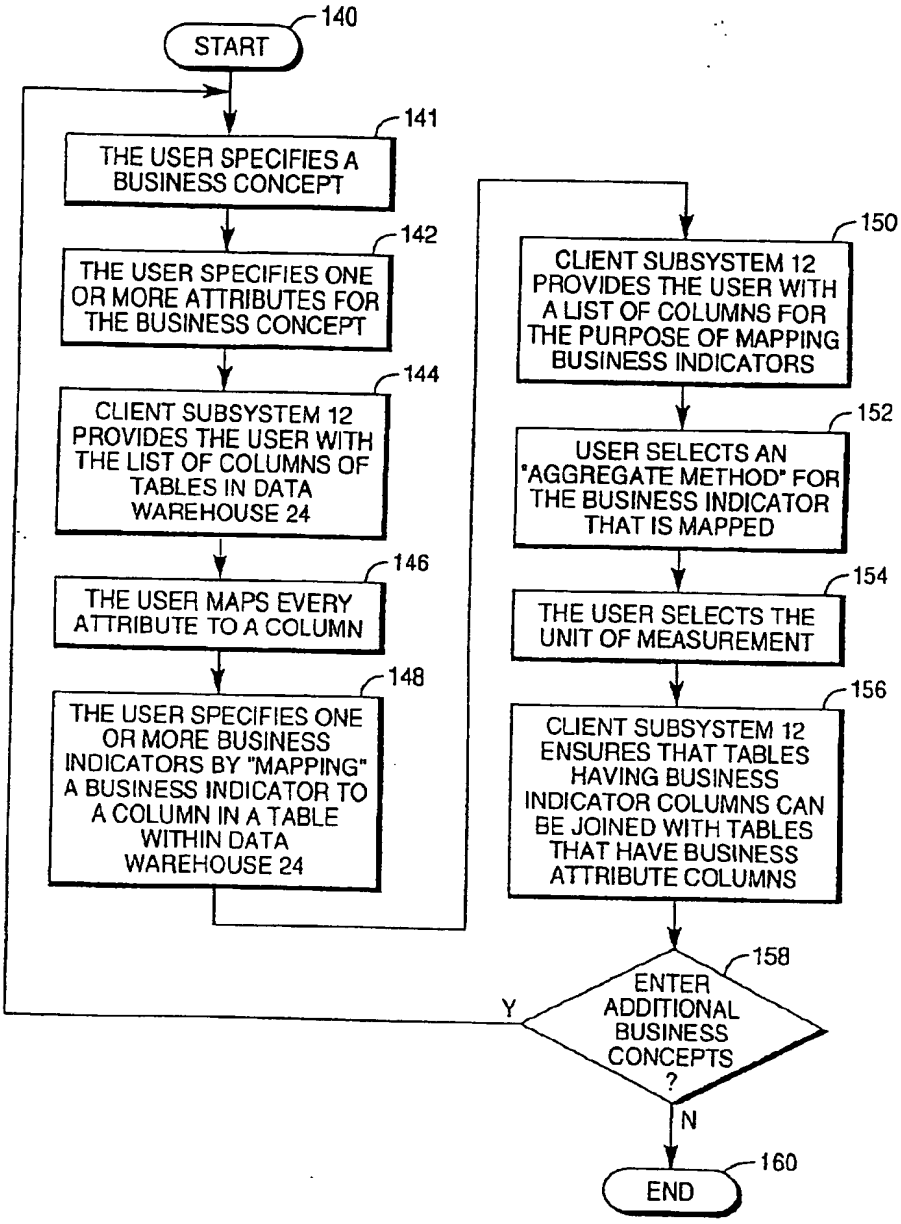

START
140
THE USER SPECIFIES A BUSINESS CONCEPT
141
THE USER SPECIFIES ONE OR MORE ATTRIBUTES FOR THE BUSINESS CONCEPT
142
CLIENT SUBSYSTEM 12 PROVIDES THE USER WITH THE LIST OF COLUMNS OF TABLES IN DATA WAREHOUSE 24
144
THE USER MAPS EVERY ATTRIBUTE TO A COLUMN
146
THE USER SPECIFIES ONE OR MORE BUSINESS INDICATORS BY "MAPPING" A BUSINESS INDICATOR TO A COLUMN IN A TABLE WITHIN DATA WAREHOUSE 24
148
CLIENT SUBSYSTEM 12 PROVIDES THE USER WITH A LIST OF COLUMNS FOR THE PURPOSE OF MAPPING BUSINESS INDICATORS
150
USER SELECTS AN "AGGREGATE METHOD" FOR THE BUSINESS INDICATOR THAT IS MAPPED
152
THE USER SELECTS THE UNIT OF MEASUREMENT
154
CLIENT SUBSYSTEM 12 ENSURES THAT TABLES HAVING BUSINESS INDICATOR COLUMNS CAN BE JOINED WITH TABLES THAT HAVE BUSINESS ATTRIBUTE COLUMNS
156
ENTER ADDITIONAL BUSINESS CONCEPTS ?
158
Y
N
END
160

FIG. 14

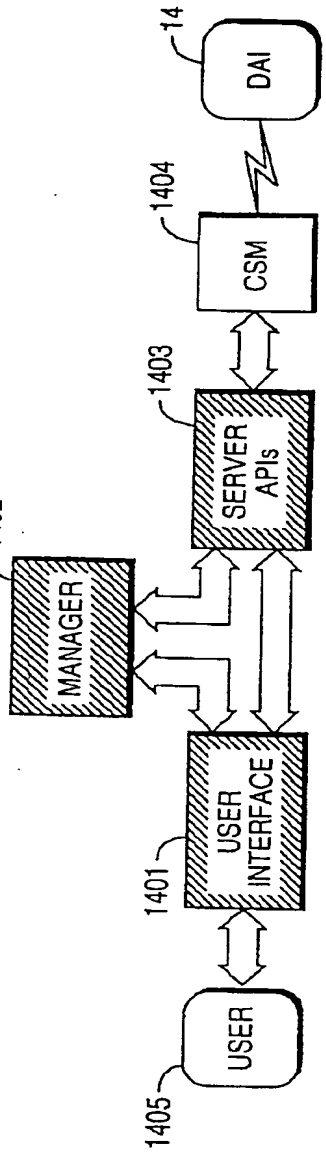

14
DAI
1404
CSM
1403
SERVER APIs
1402
MANAGER
1401
USER INTERFACE
USER
1405

FIG. 15

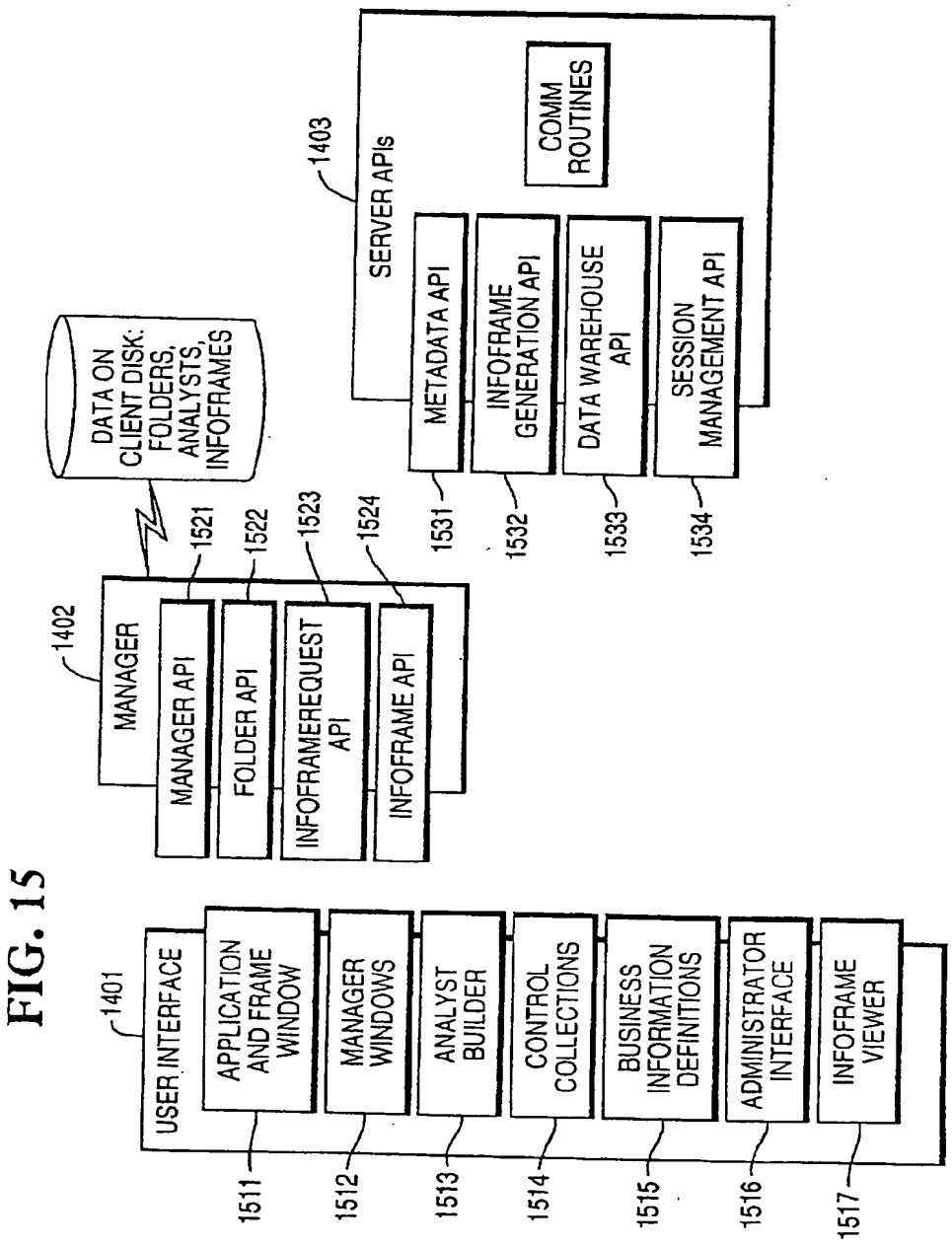

1401
USER INTERFACE
APPLICATION AND FRAME WINDOW
MANAGER WINDOWS
ANALYST BUILDER
CONTROL COLLECTIONS
BUSINESS INFORMATION DEFINITIONS
ADMINISTRATOR INTERFACE
INFOFRAME VIEWER
1511
1512
1513
1514
1515
1516
1517
1402
MANAGER
MANAGER API
FOLDER API
INFOFRAMEREQUEST API
INFOFRAME API
1521
1522
1523
1524
DATA ON CLIENT DISK: FOLDERS, ANALYSTS, INFOFRAMES
1403
SERVER APIs
METADATA API
INFOFRAME GENERATION API
DATA WAREHOUSE API
SESSION MANAGEMENT API
COMM ROUTINES
1531
1532
1533
1534

FIG. 16

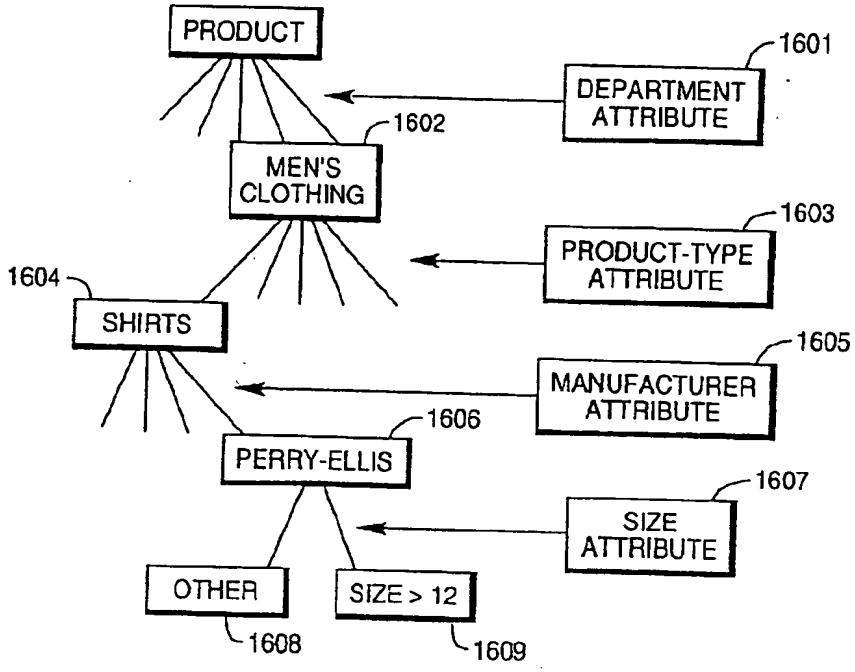

PRODUCT
1601
DEPARTMENT ATTRIBUTE
1602
MEN'S CLOTHING
1603
PRODUCT-TYPE ATTRIBUTE
1604
SHIRTS
1605
MANUFACTURER ATTRIBUTE
1606
PERRY-ELLIS
1607
SIZE ATTRIBUTE
OTHER
SIZE > 12
1608
1609

FIG. 17

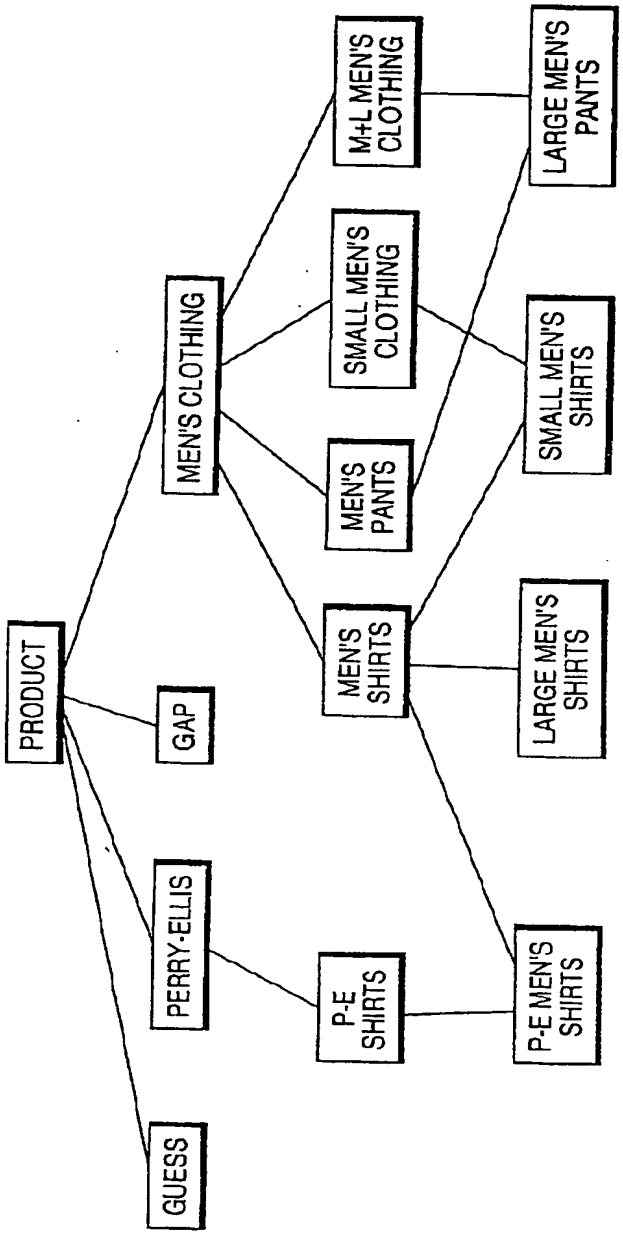

PRODUCT
GAP
GUESS
PERRY-ELLIS
MEN'S CLOTHING
MEN'S SHIRTS
P-E SHIRTS
M+L MEN'S CLOTHING
SMALL MEN'S CLOTHING
MEN'S PANTS
LARGE MEN'S PANTS
SMALL MEN'S SHIRTS
LARGE MEN'S SHIRTS
P-E MEN'S SHIRTS

FIG. 18

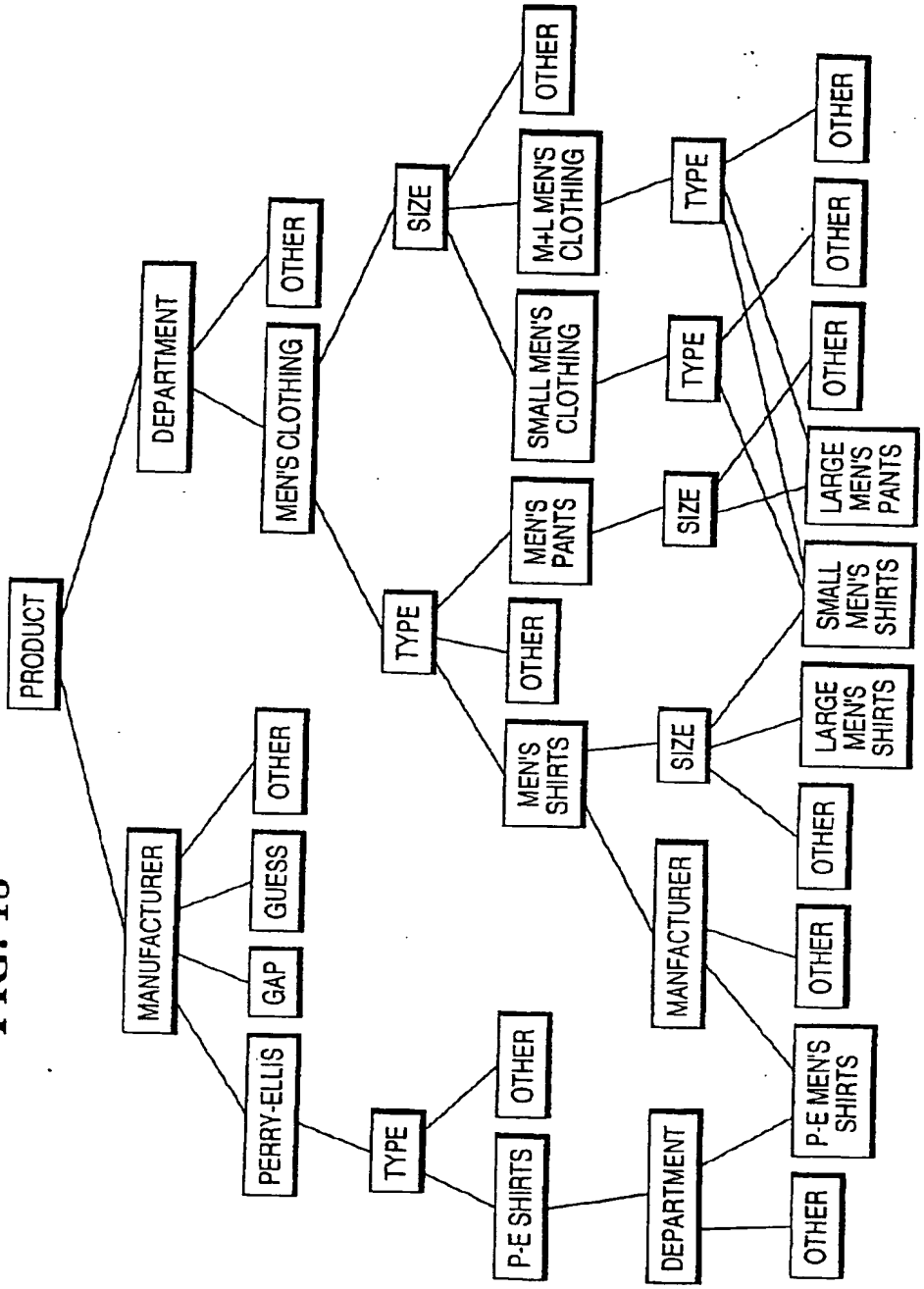

PRODUCT
DEPARTMENT
MANUFACTURER
MEN'S CLOTHING
OTHER
PERRY-ELLIS
GAP
GUESS
OTHER
SIZE
TYPE
OTHER
M+L MEN'S CLOTHING
SMALL MEN'S CLOTHING
MEN'S PANTS
OTHER
MEN'S SHIRTS
TYPE
SIZE
TYPE
SIZE
OTHER
OTHER
OTHER
LARGE MEN'S PANTS
SMALL MEN'S SHIRTS
LARGE MEN'S SHIRTS
OTHER
TYPE
OTHER
P-E SHIRTS
DEPARTMENT
MANFACTURER
OTHER
OTHER
P-E MEN'S SHIRTS

FIG. 19 25/33

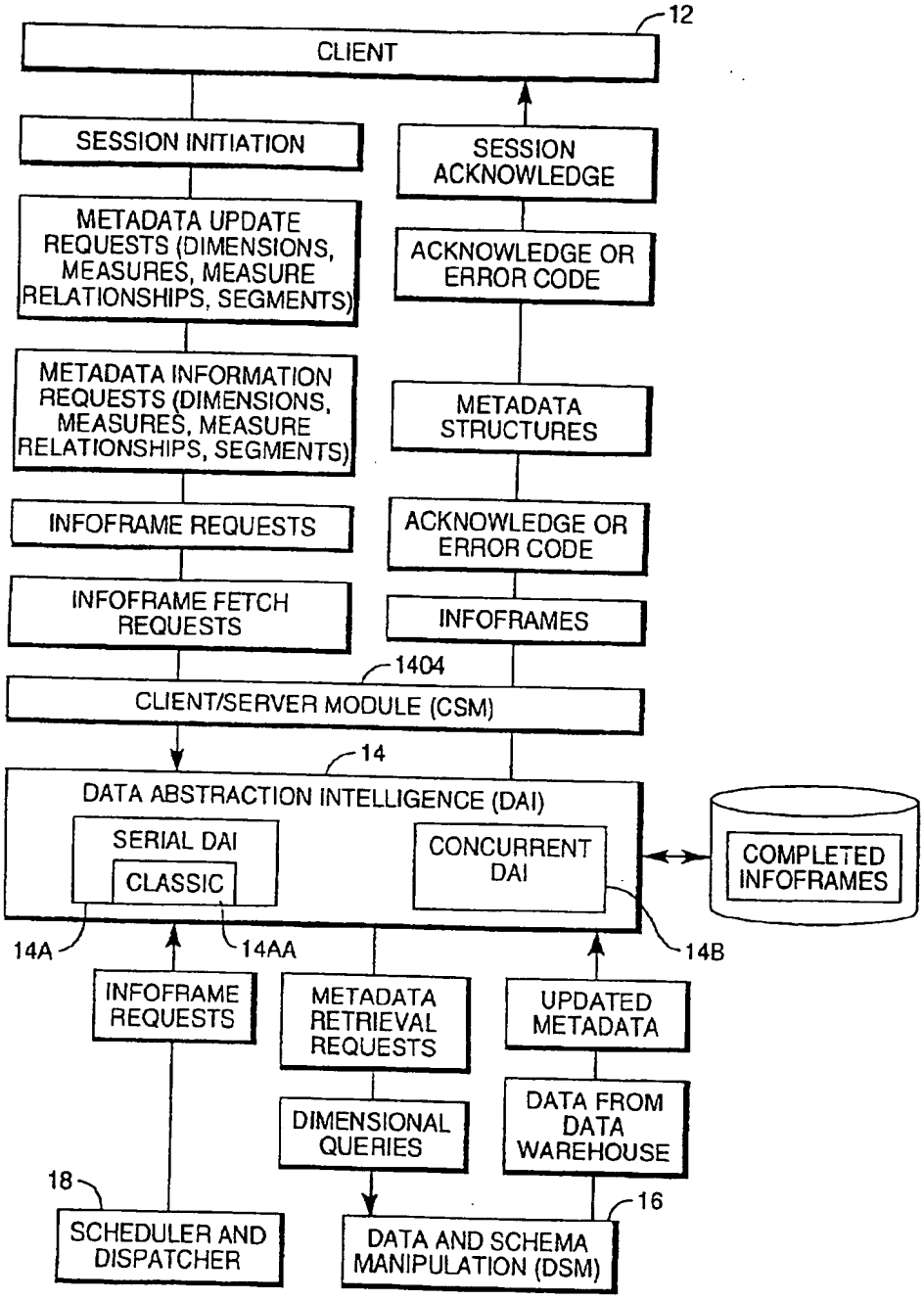

12
CLIENT
SESSION INITIATION
SESSION ACKNOWLEDGE
METADATA UPDATE REQUESTS (DIMENSIONS, MEASURES, MEASURE RELATIONSHIPS, SEGMENTS)
ACKNOWLEDGE OR ERROR CODE
METADATA INFORMATION REQUESTS (DIMENSIONS, MEASURES, MEASURE RELATIONSHIPS, SEGMENTS)
METADATA STRUCTURES
INFOFRAME REQUESTS
ACKNOWLEDGE OR ERROR CODE
INFOFRAME FETCH REQUESTS
INFOFRAMES
1404
CLIENT/SERVER MODULE (CSM)
14
DATA ABSTRACTION INTELLIGENCE (DAI)
SERIAL DAI
CLASSIC
CONCURRENT DAI
COMPLETED INFOFRAMES
14A
14AA
14B
INFOFRAME REQUESTS
METADATA RETRIEVAL REQUESTS
UPDATED METADATA
DIMENSIONAL QUERIES
DATA FROM DATA WAREHOUSE
18
16
SCHEDULER AND DISPATCHER
DATA AND SCHEMA MANIPULATION (DSM)

FIG. 20

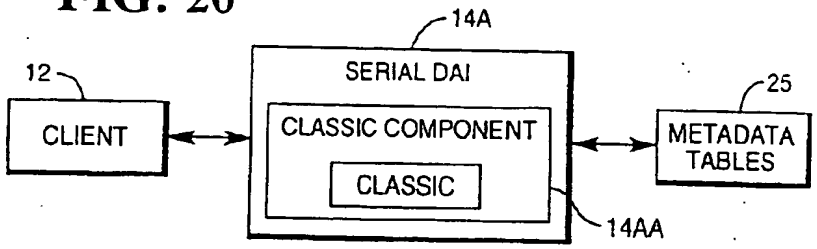


FIG. 21

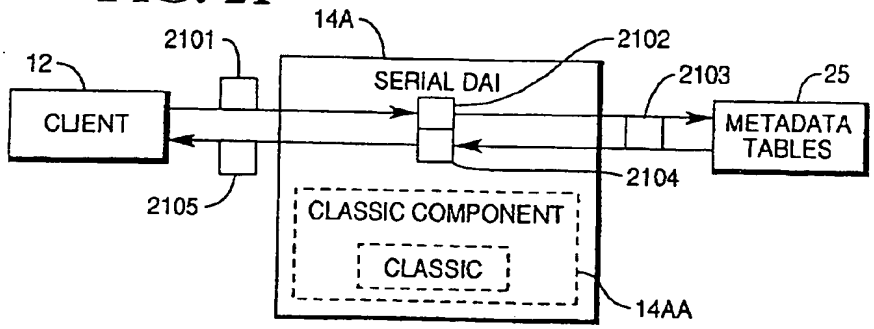
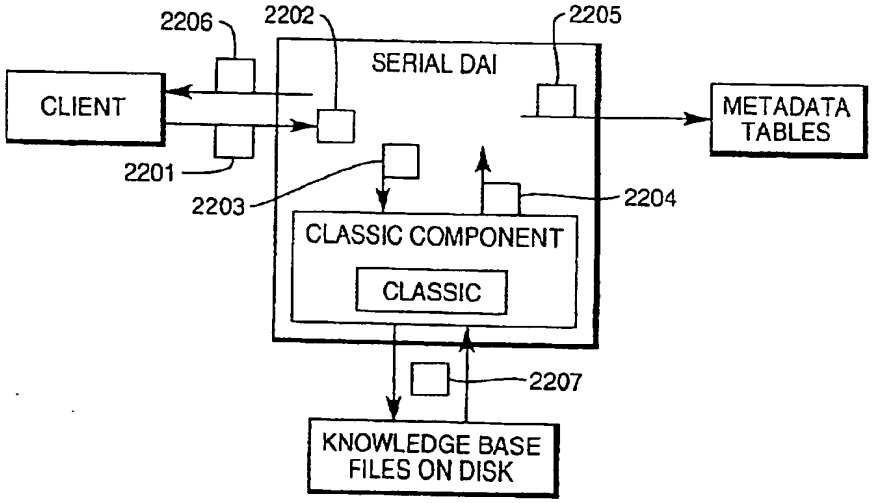


FIG. 22



27/33

FIG. 23

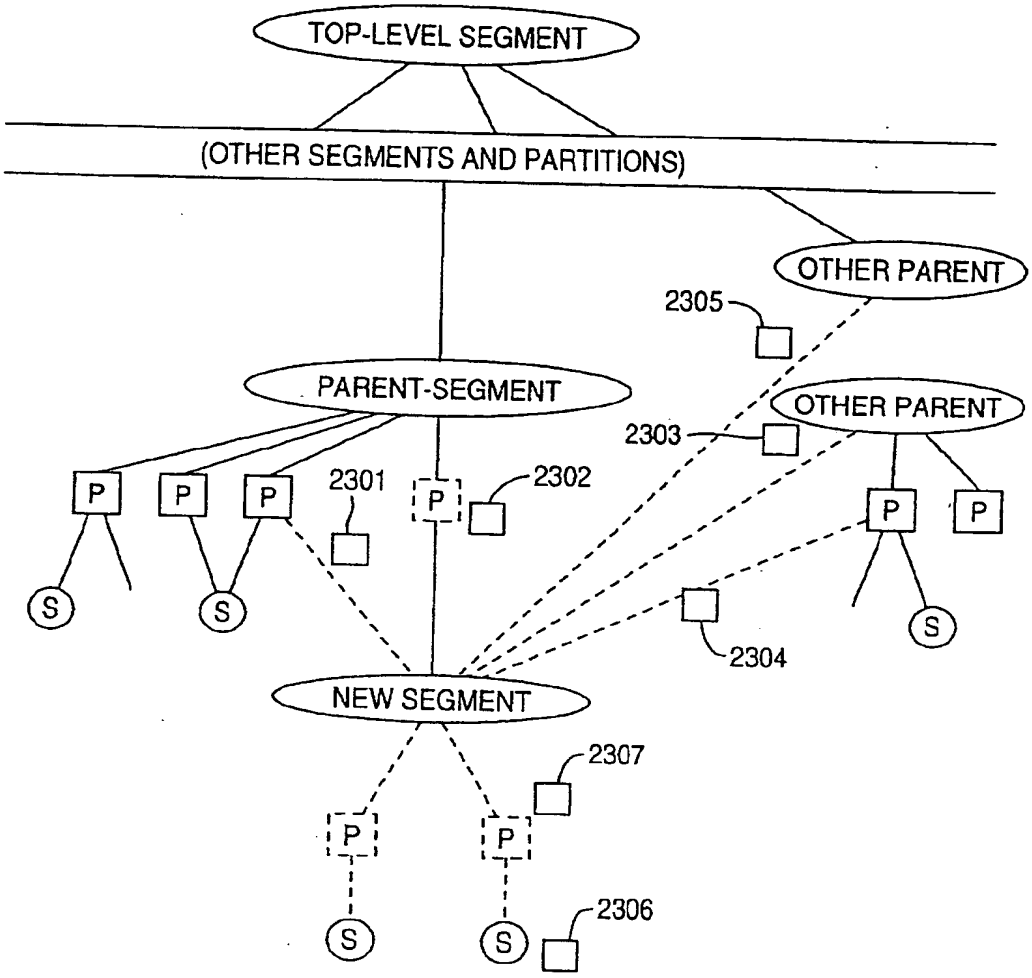

TOP-LEVEL SEGMENT
(OTHER SEGMENTS AND PARTITIONS)
OTHER PARENT
2305
PARENT-SEGMENT
OTHER PARENT
2303
P
P
P
2301
P
2302
P
P
S
S
S
2304
NEW SEGMENT
2307
P
P
S
S
2306

FIG. 24

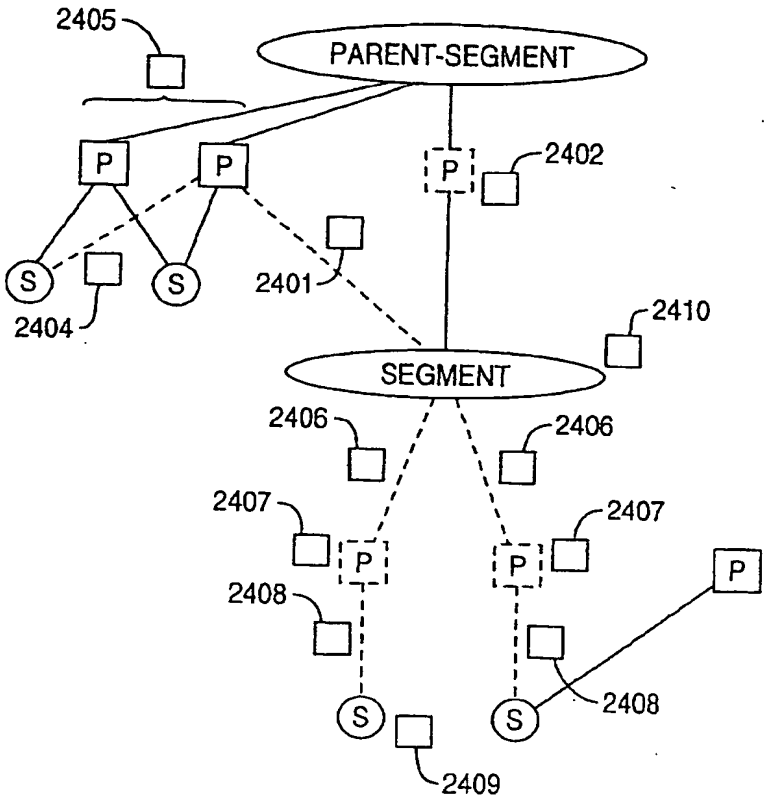

2405
PARENT-SEGMENT
P
P
P
2402
S
S
2401
2404
2410
SEGMENT
2406
2406
2407
2407
P
P
P
2408
2408
S
S
2409

FIG. 25

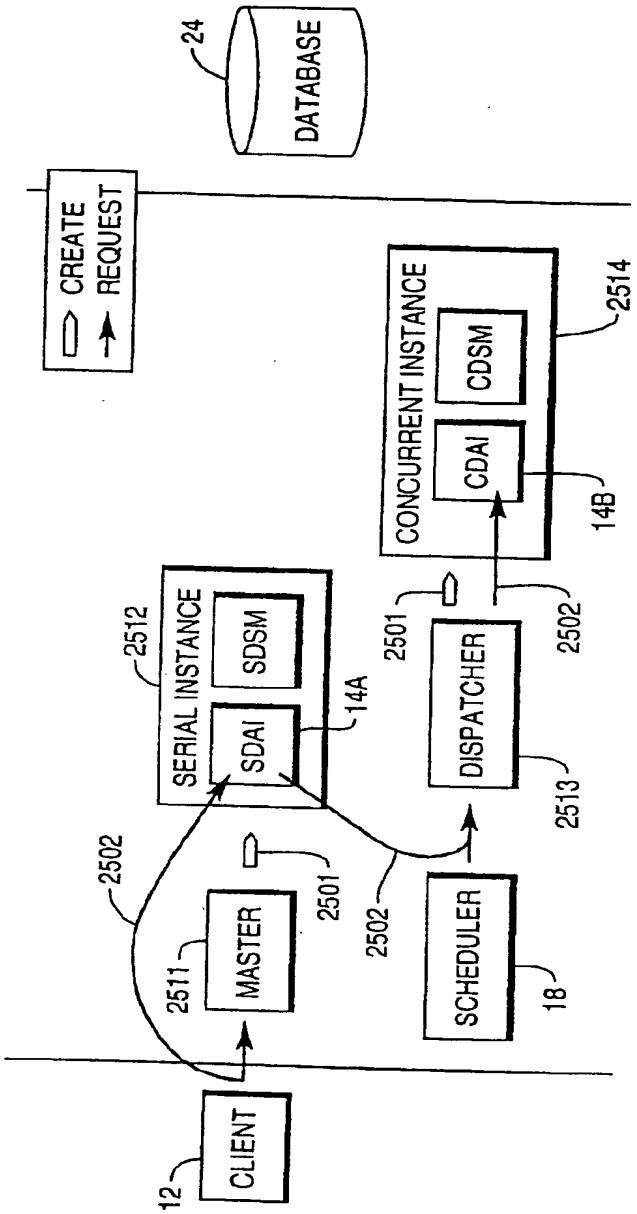

24
DATABASE
CREATE
REQUEST
CONCURRENT INSTANCE
CDSM
CDAI
2514
14B
2501
2502
DISPATCHER
2513
SERIAL INSTANCE
SDSM
SDAI
2512
14A
2502
2501
2502
MASTER
2511
SCHEDULER
18
CLIENT
12

FIG. 26

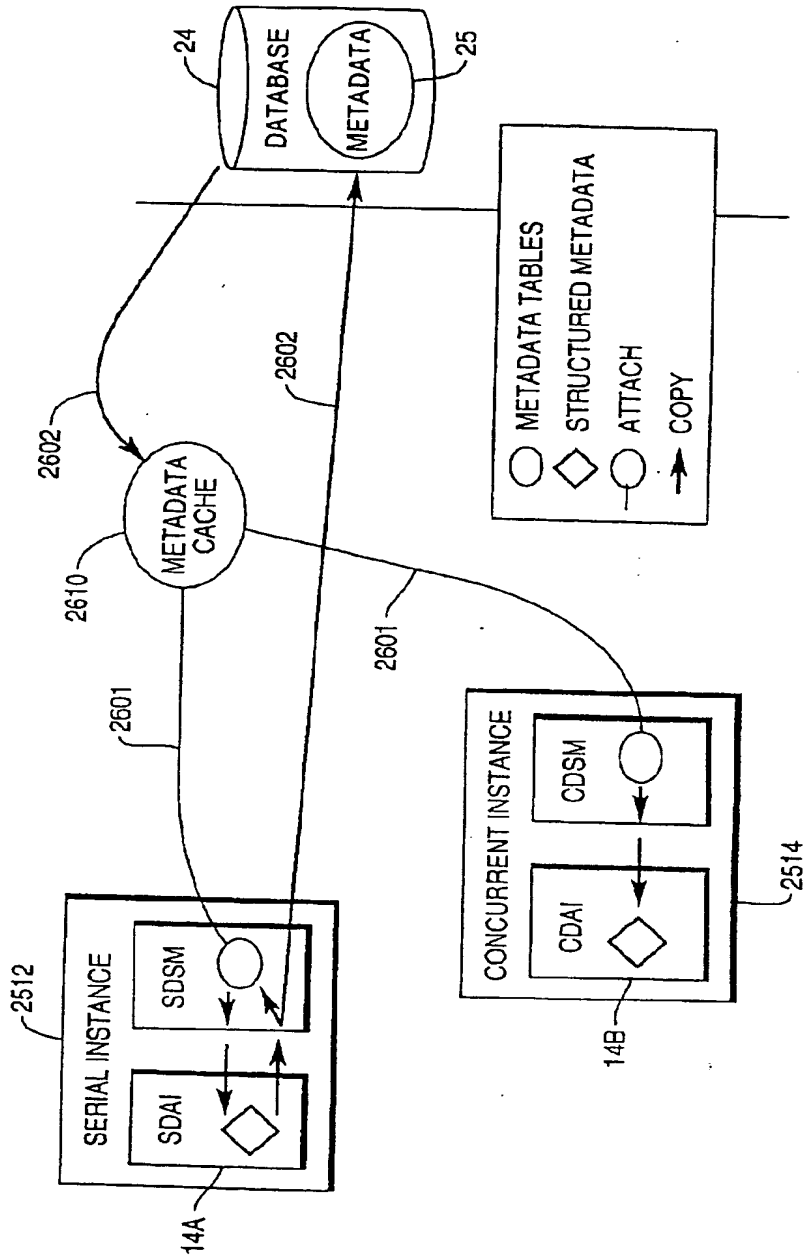

DATABASE
METADATA
24
25
METADATA TABLES
STRUCTURED METADATA
ATTACH
COPY
2602
2602
METADATA CACHE
2610
2601
2601
SERIAL INSTANCE
SDSM
SDAI
2512
14A
CONCURRENT INSTANCE
CDSM
CDAI
2514
14B

FIG. 27

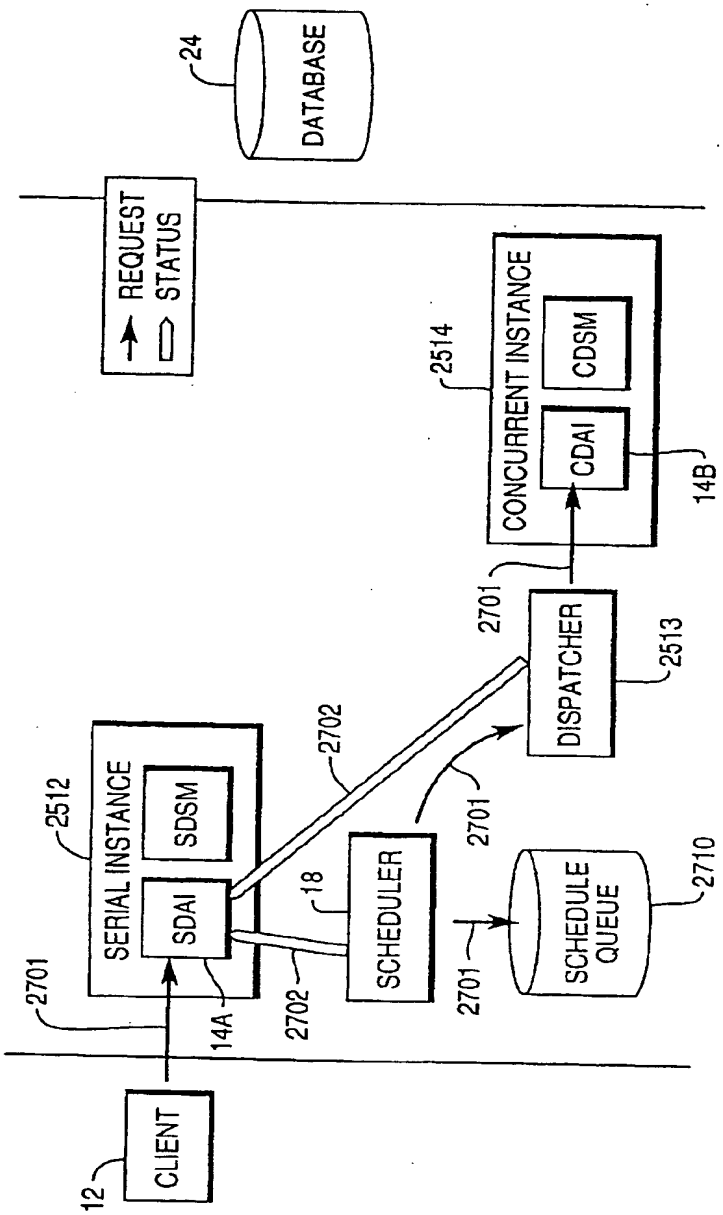

24
DATABASE
REQUEST
STATUS
2514
CONCURRENT INSTANCE
CDSM
CDAI
14B
2701
DISPATCHER
2513
2702
2512
SERIAL INSTANCE
SDSM
SDAI
18
SCHEDULER
2701
SCHEDULE QUEUE
2710
2701
2702
14A
2701
CLIENT
12

FIG. 28

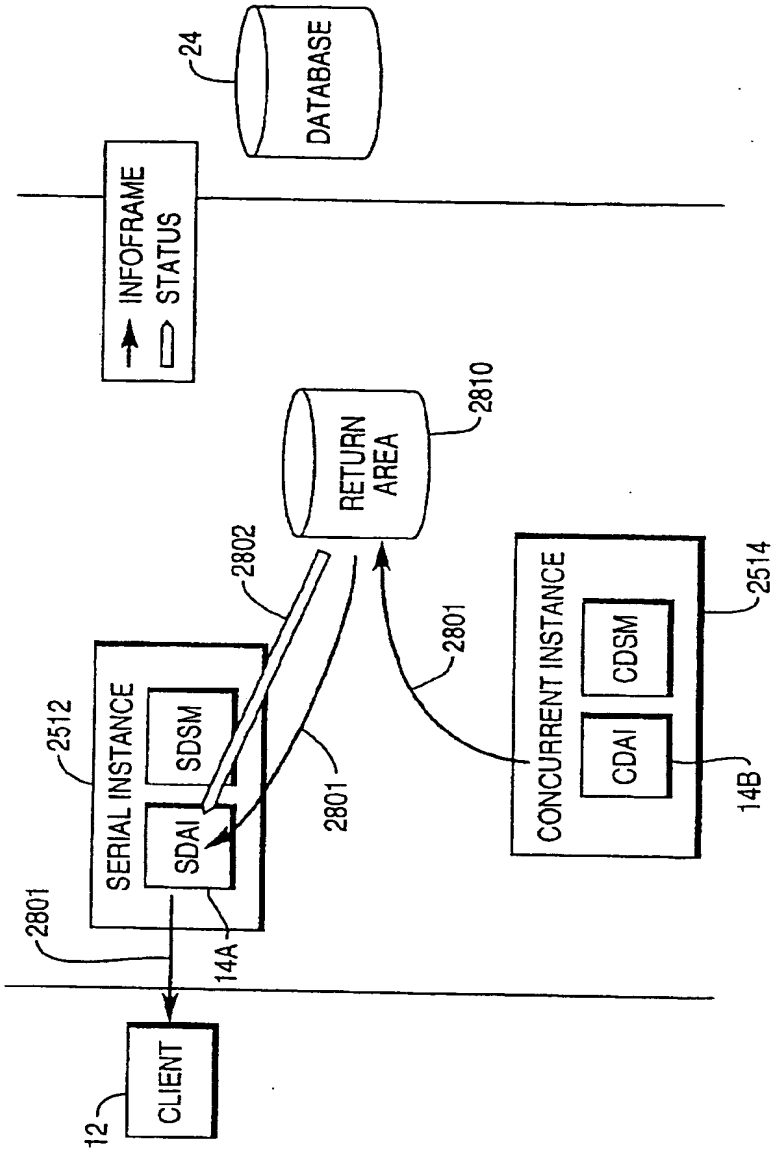
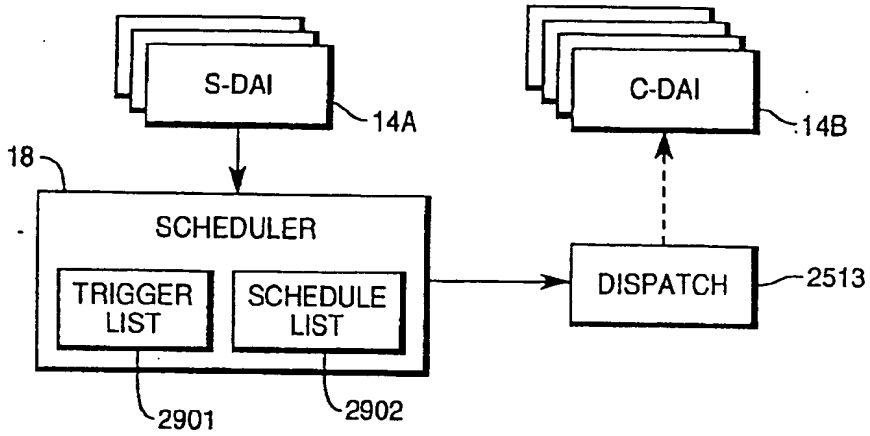

24
DATABASE
INFOFRAME
STATUS
2810
RETURN AREA
2802
2801
2514
CONCURRENT INSTANCE
CDSM
CDAI
14B
2512
SERIAL INSTANCE
SDSM
SDAI
2801
14A
CLIENT
12

FIG. 29


S-DAI
14A
C-DAI
14B
18
SCHEDULER
TRIGGER LIST
SCHEDULE LIST
DISPATCH
2513
2901
2902

A system for performing intelligent analysis, segmentation and partition of a database based upon attributes associated with the data in the database are provided. A report may be generated which allows a user to make decisions, without requiring the user to understand or interpret data itself. A database computer includes a database containing the data. The data includes a collection of information about an enterprise of the user. A server computer is coupled to the database computer and executes a database management program. A client computer is coupled to the server and executes an application program. The application program allows a user to define predetermined data types, to define relationships between the data types, to define parameters for the report, to define a method of analysis for the report, and to create the report. The report summarizes the data in terms of the data types, the data relationships and the method of analysis.

2 Representative Drawing
Fig. 1